UNIVERSIDADE FEDERAL DE SANTA CATARINA - UFSC

Solução computacional para classificação e sumarização de polaridade de comentários em português

Irenio Lima Jesus de Aragão
Nildo Wilpert Júnior

Florianópolis - SC
2018/1

UNIVERSIDADE FEDERAL DE SANTA CATARINA - UFSC
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CURSO DE SISTEMAS DE INFORMAÇÃO

Solução computacional para classificação e sumarização de polaridade de comentários em português

Irenio Lima Jesus de Aragão
Nildo Wilpert Júnior

Orientador: Prof. Dr. Elder Rizzon Santos

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Sistemas de Informação.

Florianópolis - SC
2018/1

Irenio Lima Jesus de Aragão
Nildo Wilpert Júnior

Solução computacional para classificação e sumarização de polaridade de comentários em português

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Sistemas de Informação.

Orientador:

Prof. Dr. Elder Rizzon Santos

Banca examinadora:

Prof. Dr. Ricardo Azambuja Silveira

Me. Murillo Lagranha Flores

AGRADECIMENTOS

Irenio Lima Jesus de Aragão agradece à:

Agradeço à Deus por direcionar minhas escolhas e decisões, me levando a viver experiências, conhecer pessoas e adquirir conhecimentos que eu nunca havia imaginado.

Agradeço à minha família pelo apoio constante em todos os meus caminhos, mesmo quando envolve a distância ou incertezas, principalmente à Daniele Lima Jesus de Aragão, que me suportou nos momentos de mau humor e auxiliou neste trabalho.

Sou grato aos meus amigos da SIB Floripa, que sempre me proporcionam momentos alegres e descontraídos, sem faltarem em momentos onde precisei de apoio. Sei que posso contar com todos em qualquer momento.

Como exemplos de pessoas excepcionais que eu não imaginava conhecer e conheci melhor durante este trabalho, agradeço ao Nildo Wilpert Júnior, à Maria Cristina D'amoreira de Amorim e à Luiza Moriggi, pela parceria, amizade, incentivo e compartilhamento dos momentos alegres e estressantes da vida acadêmica.

Agradeço também aos professores e colegas da UFSC, pelo conhecimento compartilhado e apoio durante o curso.

Nildo Wilpert Júnior agradece à:

Agradeço primeiramente à minha família, meus pais e minhas irmãs, pois sem vocês não poderia ter chegado onde estou. A minha noiva, Maria Cristina D'amoreira de Amorim, um agradecimento especial por toda ajuda incondicional, todas as palavras de incentivo e toda a paciência ao longo desta caminhada. Sem você a conclusão desta etapa teria sido muito mais difícil.

Agradeço também a todos os meus amigos, colegas e professores, que de muitas formas me ajudaram durante toda a graduação e durante o desenvolvimento deste trabalho.

RESUMO

Analisar a qualidade de um produto ou serviço levando em consideração os comentário de outros consumidores se tornou uma prática comum nos dias de hoje. Realizar esta análise de forma consolidada, eficiente e confiável através de um conjunto de comentários é, muitas vezes, impraticável. Programas capazes de auxiliar os usuários a realizar tal tarefa se fazem necessários. Tanto para auxiliar consumidores, quanto organizações, que veem no *feedback* de seus clientes um meio para melhorar seus produtos e serviços. O objetivo deste trabalho é propor uma solução computacional capaz de realizar a análise de um conjunto de comentários, em português do Brasil, com o intuito de levantar os principais pontos positivos e negativos sobre um determinado produto, apresentando-os em forma de sumário estatístico. Para tal propósito, realizou-se um estudo sobre as técnicas de inteligência artificial e processamento de linguagem natural empregadas na solução deste tipo de problema, e um protótipo foi implementado para a solução proposta utilizando tais técnicas. Este documento apresenta o relato sobre os estudos realizados, a solução proposta, a implementação do protótipo e os resultados dos testes, além do código fonte de tal solução.

Palavras-chave: inteligência artificial, processamento de linguagem natural, análise de sentimentos, mineração de opiniões, sumarização de opiniões, avaliações de produtos.

ABSTRACT

Analyze the quality of a product or service taking into consideration the comments of others customers has become a common practice nowadays. Doing analysis on a consolidated, efficient and trustworthy way using a large number of comments is much times impracticable. Computer programs capable of helping users to do this task became necessary, both for customers, as well as organizations, who see feedback of their customers as a way to improve their products and services. The goal of this work is to propose a software capable of analyzing a collection of Brazilian Portuguese comments about one product to show the main positive and negative points about it, presenting them as a statistical summary. For this purpose, a study was carried out on artificial intelligence and natural language processing techniques used in the solution of this type of problem, and a prototype was implemented for the proposed solution using such techniques. This document presents the report on the studies carried out, the proposed solution, the implementation of the prototype and the results of the tests, besides the source code of such solution.

Key words: artificial intelligence, natural language processing, sentiment analysis, opinion mining, opinion summarization, product ratings.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API - Application Programming Interface – Interface de Programação de Aplicações

IA - Inteligência Artificial

KNN - *K-Nearest Neighbors* - K vizinhos mais próximos

NLP - *Natural Language Processing*

NLTK - *Natural Language Toolkit*

PLN - Processamento de linguagem natural

POS - *Part of Speech* - Parte do discurso

SVMs - *Support Vector Machines* - Máquinas de vetores de suporte

VISL - *Visual Interactive Syntax Learning* - Aprendizagem de sintaxe interativa visual

SUMÁRIO

1. INTRODUÇÃO

Inteligência artificial pode ser definida como o ramo da ciência da computação que se ocupa da automação do comportamento inteligente (LUGER, 2014, p. 1). Nos anos 50, quando os primeiros conceitos de Inteligência Artificial (IA) surgiram, o paradigma explorado era o simbologista. Que consiste no desenvolvimento de modelos utilizando representações explícitas da informação contendo símbolos organizados com uma sintaxe específica. Nos anos 80 foi proposto o paradigma conexionista, que consiste em um modelo massivamente paralelo, composto por um grande número de simples elementos de processamento interconectados. Este modelos tem como grande vantagem a capacidade de resolver problemas que necessitam de flexibilidade e robustez (SUN, 2000).

O Processamento de Linguagem Natural (PLN), segundo Chowdhury (2003, p. 51) é um campo da Inteligência Artificial (IA) de pesquisa e aplicação que explora como computadores podem ser usados para entender e manipular textos ou falas com linguagens naturais para alguma utilidade.

No seu surgimento, PLN estava intrinsecamente relacionado com o paradigma simbologista e envolvia a codificação de um grande conjunto de regras. Mas com o passar dos anos e com os avanços das pesquisas, técnicas conexionistas passaram a ser empregadas a fim de melhorar os sistemas já existentes e propor novos. Estas técnicas se apoiam em aprendizado de máquina, onde corporas são utilizados para ensinar aos algoritmos tais regras. Corporas (singular *corpus*), são grandes conjuntos estruturados de documentos ou sentenças individuais escritos usando as regras e vocabulários de uma linguagem (WIKIPEDIA, 2016).

Atualmente grande parte da população está preocupada com a qualidade dos produtos e serviços que estão consumindo. Em geral, os consumidores desejam conhecer quais características fazem o produto ser considerado bom ou ruim de acordo com a opinião dos que já o utilizam.

Dados do Portal do Consumidor (CONSUMIDOR, 2015; CONSUMIDOR, 2016) mostram que as reclamações que foram registradas por consumidores e

resolvidas por parte da empresa no primeiro trimestre de 2016 aumentaram em 40% em relação ao mesmo período de 2015.

O crescimento na utilização de sites de avaliação e reclamação resulta no aumento dos dados e registros disponibilizados por tais ferramentas, o que dificulta cada vez mais a análise consolidada, eficiente e confiável, sem a subjetividade dos registros individuais, por parte do consumidor. Do outro lado, para as organizações é difícil utilizar este crescente volume de dados textuais para melhorar seus negócios.

Percebeu-se, então, a necessidade de desenvolver uma ferramenta computacional capaz de analisar as avaliações e apresentar de forma centralizada os principais pontos positivos e negativos de um produto pesquisado. A eficiência de uma análise deste tipo depende da correta compreensão de informações textuais, identificando características de avaliação de um produto e classificando-as de forma qualitativa. Neste trabalho, o processamento de linguagem natural será utilizado com o objetivo de atender a esta necessidade.

1.1 OBJETIVO GERAL

Desenvolver uma solução computacional utilizando técnicas de PLN para análise de textos em português do Brasil, provenientes de sites de avaliação e/ou reclamação de produtos, com o intuito de auxiliar consumidores no momento da escolha, e organizações na melhoria dos seus negócios.

1.2. OBJETIVOS ESPECÍFICOS

1. Analisar técnicas de IA pertinentes a PLN e elencar as mais apropriadas para a implementação da solução proposta;
2. Desenvolver uma ferramenta, utilizando os algoritmos estudados em 1, capaz de analisar um conjunto de comentários, escritos em português do Brasil, e sumarizar suas polaridades, contando as avaliações positivas e negativas sobre cada característica comentada;
3. Testar a ferramenta desenvolvida em 2;
4. Avaliar os resultados obtidos em 3 e propor melhorias para a ferramenta desenvolvida em 2.

2. REFERENCIAL TEÓRICO

2.1. PROCESSAMENTO DE LINGUAGEM NATURAL

Processamento de Linguagem Natural (PLN) pode ser definido como a área da Ciência da Computação responsável por estudar e propor técnicas para o desenvolvimento de programas de computador que sejam capazes de analisar, reconhecer e/ou gerar conteúdo oral ou escrito em linguagens humanas ou linguagens naturais (VIEIRA; LOPES, 2010).

Linguagens naturais são aquelas usadas por humanos para, entre outras funções, a comunicação. Podemos citar como exemplo de linguagens humanas o português e o inglês, mas existem diversas outras línguas. No ano de 2016, ainda podem ser encontradas no mundo cerca de 7,097 línguas (LEWIS; GARY; FENNIG, 2016), e todas têm uma característica comum: a ambiguidade.

Na linguística, ambiguidade é a possibilidade de uma expressão (palavra, frase, sentença) ter mais de um significado ou ser entendida de diferentes maneiras. Isso pode ocorrer em qualquer um dos níveis de análise contidos em PLN (K; ANTO, 2007). Para diferentes níveis existem diferentes técnicas de desambiguação, por exemplo: part of speech tagging, tratada na seção 2.1.1.1, para o processamento morfológico; ou word sense disambiguation, para o semântico (BHATTACHARYYA, 2012). Estas técnicas permitem uma análise mais acurada do significado das frases e do discurso como um todo. A ambiguidade diferencia o PLN do processamento de linguagens de programação de computador, para as quais existem definições formais que, entre outras coisas, evitam ambiguidades (VIEIRA; LOPES, 2010). A informalidade e os possíveis erros de escrita ou fala encontrados no uso de linguagens naturais são tratados no PLN através de conhecimentos da área de linguística, permitindo maior aproveitamento do conteúdo do texto, realizando correções e desambiguações, identificando sinônimos para as palavras e classificando entidades e seus relacionamentos (ARANHA; PASSOS, 2006).

De acordo com Vieira e Lopes (2010), além da variedade de linguagens classificadas como naturais, existem, para cada uma, possíveis variações relacionadas com a forma em que são utilizadas e o propósito de tal uso. Assim,

diferentes métodos ou técnicas computacionais podem atender às diferentes necessidades de análise e interpretação de linguagem natural. Há distinção, por exemplo, entre técnicas aplicadas à linguagem falada e à linguagem escrita, como também entre técnicas para processar linguagens de diferentes contextos de aplicação, como as áreas científica, jornalística, literária, pedagógica, cultural etc. Por esta razão, a grande maioria das aplicações de PLN são focadas em um tipo específico de linguagem.

Desde o início da década de 1990, as pesquisas e o uso do PLN tem se direcionado mais para o tratamento de textos do que para o tratamento da fala. Isso devido ao crescimento no volume de conteúdos textuais, resultante do crescimento da internet e de sua inserção nos ambientes acadêmico, cultural e empresarial. Com este direcionamento, novas pesquisas levaram ao desenvolvimento de corporas anotados sintaticamente, que são conjuntos de textos sobre domínios de conhecimento específicos, com cada palavra identificadas de acordo com sua função sintática (VIEIRA; LOPES, 2010).

Para reconhecer, interpretar, e responder apropriadamente ao dinamismo das situações e contextos em que a linguagem é encontrada, não apenas em gêneros formais e bem escritos, são essenciais ao PLN níveis adicionais de interpretação, além da semântica padrão, que incorporam todos os conjuntos de características que transmitem significado com base em pistas linguísticas e paralinguísticas usadas pelos seres humanos em suas interações de comunicação social, seja na fala ou na escrita. Sinais como a marcação de ênfase, e outros símbolos não léxicos, os níveis de inflexão e de energia na voz e efeitos pragmáticos formalizados no nível léxico tendem a ser menos governados formalmente por regras de sintaxe ou papéis semânticos. Além desses sinais, a interação social pode se utilizar de dispositivos de comunicação, tais como delicadeza, dúvida, ou sarcasmo, bem como variar a quantidade de detalhes ou nível de clareza necessária para o contexto. Essas sutilezas constroem informação contextual e conhecimentos específicos na comunicação, que dirigem a forma como uma mensagem deve ser entendida e, se necessário, respondida (LIDDY et al., 2003).

Segundo Jurafsky e Martin (2009) PLN pode ser dividido em seis categorias distintas:

Análise fonética e fonológica, que trata da relação entre as palavras e seus sons;

Análise morfológica, responsável pelo estudo da construção das palavras a partir de componentes dotados de significado e da classificação das mesmas em categorias morfológicas;

Análise sintática, que considera o relacionamento entre as palavras, analisando a estrutura para formação das sentenças;

Análise semântica, focada no significado das palavras e das sentenças;

Análise pragmática, que estuda como as sentenças são estruturadas a fim de transmitir uma mensagem entre um locutor e um receptor em um determinado contexto;

Análise de discurso, cujo objetivo é entender como as sentenças se relacionam para formar estruturas mais complexas.

No restante desta seção cada uma das categorias citadas será tratada em mais detalhes, com exceção da análise fonética e fonológica, tendo em vista que não se aplica ao escopo deste trabalho.

2.1.1 Análise morfológica

De acordo com Jurafsky e Martin (2009) todas as linguagens humanas, sejam elas através de sinais, fala ou escrita, são formadas de um mesmo elemento básico, as palavras. Por este motivo todo sistema de processamento de linguagem natural deve possuir vasto conhecimento sobre elas.

Segundo Ferreira (2008, p.564), morfologia é “o estudo da estrutura e formação das palavras”. Em linguística, morfologia é a área responsável por estudar como as palavras são formadas a partir de morfemas (JURAFSKY; MARTIN, 2009). Um morfema pode ser definido como o “elemento que confere o aspecto gramatical ao semantema, relacionando-o na oração e delimitando sua função e seu significado. Ex.: os afixos” (FERREIRA, 2008, p.564). Ou também como o “elemento linguístico mínimo que possui significado” (FERREIRA, 2008, p.564).

A separação em morfemas permite reduzir o tamanho do dicionário da língua a ser analisada. Isso se dá através do tratamento de palavras no plural (Ex.: celulares, formada pelos morfemas celular e -es) e, em alguns casos, devido a

possibilidade de agrupar palavras com o mesmo radical (Ex.: livro e livrinho, cuja radical é livr), à este último processo dá-se o nome em inglês *stemming*.

Na etapa de análise morfológica também é realizada a tarefa de etiquetamento das palavras em classes gramaticais, comumente citada utilizando-se o termo em inglês *Part of Speech* (POS) *tagging*. Está é uma etapa de grande importância para os sistemas de PLN e por este motivo será tratada em maiores detalhes na seção 2.1.1.1.

2.1.1.1 POS *tagging*

POS *tagging* é o processo de associação de cada uma das palavras de um texto com suas classes gramaticais, baseado tanto na sua definição quanto em seu contexto, ou seja, a relação com as palavras adjacentes (BACKER, 2014; JURAFSKY e MARTIN, 2009). A etapa de POS é a primeira a inserir informação ao texto e com isso provê a base para as análises subsequentes. Por este motivo é de extrema importância nos sistemas de processamento de linguagem natural. Um sistema que possua uma análise morfológica falha será, consequentemente, ineficaz. Está é uma etapa de particular importância para os sistemas de análise de sentimentos pois determinadas classes são consideradas pontos chave para a definição de polaridade.

Segundo Jurafsky e Martin (2009) esta não é uma tarefa trivial, principalmente pois um grande número de palavras é ambígua, isto é, pode possuir mais de uma classe gramatical dependendo do contexto onde se encontra. POS é, portanto, uma das primeiras formas de desambiguação possível em PLN.

A maioria dos algoritmos de POS se encaixa nas categorias baseado em regras (*rule-based* em inglês) ou estocástico (do inglês *stochastic*). O primeiro, comumente, faz uso de um grande conjunto de dados anotados manualmente para tratar a desambiguação. Já o segundo, geralmente, utiliza um corpus de treinamento para calcular a probabilidade de uma palavra pertencer à uma classe dado um determinado contexto. Um exemplo clássico de algoritmo de POS estocástico é o *Hidden Markov Model (HMM) tagger*.

Ao longo dos anos foram criados diversas ferramentas para realizar tal tarefa. Devido às características das linguagens naturais, geralmente estas tratam de um

idioma específico. O sitio LX_Center[1], solução proprietária, e o sitio Linguateca[2], são exemplos de ferramentas capazes de processar textos em português, mas também existem ferramentas multi idiomas como, por exemplo, o NLTK *toolkit*[3], desenvolvido na linguagem Python, que disponibiliza funcionalidades para o português do Brasil, entre outras.

2.1.2 Análise sintática

"Se palavras são a base no processamento de discurso e linguagem natural, sintaxe é o esqueleto"[4] (JURAFSKY; MARTIN, 2009, p.283, tradução nossa). Sintaxe, segundo Ferreira (2008, p.742), é a "parte da gramática que estuda a disposição das palavras na frase e das frases no discurso".

A análise sintática é a responsável por agrupar as palavras em estruturas que mostram como elas estão relacionadas entre si, convertendo a lista de palavras, resultante da análise morfológica, em uma estrutura que define as unidades de significado representadas por aquela lista. Esse tipo de análise, segundo Cordeiro (2016, p. 124), "possibilita resolver alguns problemas de ambiguidade, onde algumas palavras podem ser classificadas diferentemente dependendo da sua função na oração".

Esse processo, chamado de análise (*parsing*, em inglês), é baseado nas regras de combinação das palavras, definidas em cada linguagem, o que permite que algumas sequências de palavras sejam rejeitadas se violarem tais regras. Por exemplo, um analisador sintático do português rejeitaria a frase "Garota a vai mercado ao".

Como resultado, a análise sintática deve encontrar os componentes que posteriormente receberão significados, no processo de análise semântica (RICH; KNIGHT, 1994).

---

[1] lxcenter.di.fc.ul.pt

[2] www.linguateca.pt

[3] www.nltk.org

[4] If words are the foundation of speech and language processing, syntax is the skeleton. (JURAFSKY; MARTIN, 2009, p.283)

2.1.3 Análise Semântica

Ferreira (2008, p.731) define semântica como o “estudo das mudanças ou transladações sofridas, no tempo e no espaço, pela significação das palavras”.

A análise semântica produz uma representação do significado da frase. De acordo com os objetos e conceitos do domínio sendo tratado, esse processo se encarrega de atribuir significado às estruturas encontradas na análise sintática (CORDEIRO, 2016). As estruturas que não puderem ser mapeadas são rejeitadas, sendo consideradas semanticamente anômalas para o universo de conhecimento em questão.

A compreensão é um processo de mapeamento de uma forma de entrada para uma forma mais útil em um contexto. Na análise semântica, a compreensão depende da definição da linguagem alvo, ou *linguagem-objeto*, para a qual se deseja mapear, o que define a estrutura representacional e o vocabulário de significado específico. A escolha da linguagem-objeto deve ser guiada pelo que será feito com as estruturas e significados após a construção, pois irá definir as restrições da construção de representações, considerando apenas os significados que fazem sentido no universo de conhecimento sendo tratado (RICH; KNIGHT, 1994).

2.1.4 Análise de discurso e Pragmática

Enquanto as etapas anteriores eram utilizadas para o processamento de palavras ou sentenças isoladas, a análise de discurso é responsável pelo processamento de grupos de sentenças relacionadas.

Devido a esta característica, a análise de discurso é responsável por interpretar a forma como pessoas e coisas são referenciadas, utilizando para isso algoritmos de resolução de referências. E também modelar a maneira que o discurso deve ser estruturado para que o ouvinte (ou leitor) seja capaz de interpretar o que está sendo transmitido.

Pragmática, por sua vez, é o estudo da relação entre a língua e contexto no qual ela é usada. Ou seja, é a aplicação do discurso em uma dada época e um certo contexto. (JURAFSKY; MARTIN, 2009).

2.2 LINGUÍSTICA DE CORPUS

A palavra “corpus” pode ser definida como “compilação de documentos ou informações relativos a uma disciplina ou um tema” (CORPUS, 2016), não se tratando de um termo exclusivamente computacional. Já a Linguística de Corpus, segundo Sardinha (2004), dedica-se à exploração da linguagem através de evidências empíricas, extraídas por meio de computador, tratando da coleta e exploração de conjuntos de dados linguísticos textuais com o propósito de servirem para a pesquisa de uma língua ou variedade linguística.

Por tratarem-se de documentos textuais, os corpora são específicos para o idioma no qual seus textos estão escritos. Alguns dos corpora disponíveis para o português do Brasil podem ser encontrados sob o projeto Floresta Sintá(c)tica[5] uma parceria entre a Linguateca (LINGUATECA, 2003) e o projeto VISL (VISL, 1996). Uma análise mais aprofundada dos corporas disponíveis para o português do Brasil pode ser encontrada em Sardinha (2004).

Nos últimos anos, técnicas de PLN baseadas em corpus tem se unido com técnicas de aprendizagem de máquina, ou mais especificamente, técnicas de mineração de dados, que se aplicam a conjuntos de dados de onde se pode inferir informação, mas que são humanamente intratáveis (VIEIRA; LOPES, 2010).

2.3 APRENDIZADO DE MÁQUINA

Mitchell (1997, p. 2, tradução nossa) usa a seguinte definição para aprendizado de máquina: “Diz-se que um programa de computador aprende de uma experiência E, com respeito a uma classe de tarefas T e medida de performance P, se sua performance nas tarefas de T, mensuradas por P, aumenta com a experiência E”[6]. Ou seja, a capacidade de aprendizado de um sistema ao passar por determinado processo de aprendizagem, é evidenciada mensurando o desempenho do sistema ao executar uma tarefa de seu domínio e comparando a performance anterior e posterior ao processo. Usualmente, aprendizado de máquina é separado

[5] http://www.linguateca.pt/Floresta/

[6] A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E. (MITCHELL, 1997, p. 2)

em três categorias primárias: aprendizado supervisionado, aprendizado não supervisionado e aprendizado semi supervisionado.

2.3.1 Aprendizado supervisionado

No aprendizado supervisionado o sistema de aprendizado recebe um conjunto de exemplos (conjunto de treinamento), onde cada exemplo, representado por suas características ou atributos, é rotulado de acordo com a categoria ao qual pertence. Como resultado, o sistema de aprendizado deve construir um modelo de uma função, chamada de função conceito (*concept function*), que permite predizer valores para novas entradas, diferentes dos exemplos vistos previamente.

Na maioria dos casos, a quantidade de exemplos utilizados no aprendizado não é suficiente para se obter uma função que modele qualquer das possíveis entradas do domínio sendo tratado. Na realidade, os sistemas de aprendizado são capazes de induzir uma função que se aproxima da função conceito, chamada de hipótese (GENTLEMAN; CAREY, 2008).

Para aprender a partir de um conjunto de exemplos, é necessário definir de que forma os exemplos serão analisados para criar um modelo e como novas entradas serão comparadas, ou aproximadas, dos exemplos já conhecidos. Para isso, diversos paradigmas foram propostos no aprendizado de máquina supervisionado e diferentes técnicas foram elaboradas.

2.3.1.1 Árvore de Decisão (Decision Trees)

No aprendizado através de árvore de decisão os dados de treinamento são utilizados para criar um modelo em forma de árvore capaz de classificar novas entradas com o menor custo possível (menor caminho).

Nesta estrutura cada nodo representa um atributo, cada aresta representa um valor de atributo e cada nodo folha representa um possível valor de classificação (positivo ou negativo, por exemplo).

Após ter o modelo da árvore de decisão criado, a função de classificação recebe um valor de entrada e retorna a classificação para ele. A classificação é obtida navegando pela árvore, avaliando os atributos encontrados no nodo e

seguindo pelo ramo que representa o valor do atributo na entrada recebida até atingir um nodo folha. Quando isso acontece, o algoritmo retorna a classificação representada pelo nodo.

2.3.1.2 *Naive Bayes*

Naive Bayes é um algoritmo de aprendizado que segue a abordagem do método bayesiano. Em tal método, a aprendizagem ocorre através do cálculo da probabilidade para dadas hipóteses, a partir de características dos exemplos observados (MITCHELL, 1997).

Segundo Mitchell (1997), as características do aprendizado bayesiano incluem:

- Flexibilidade para que cada exemplo no conjunto de dados altere de forma incremental a probabilidade de uma hipótese estar ou não correta. Diferente de algoritmos que descartam uma hipótese em função da inconsistência de um único exemplo.
- Possibilidade de combinar probabilidade baseada nas características de cada exemplo com a distribuição de probabilidade sobre os dados observados.
- Capacidade de tratar hipóteses que fazem predições probabilísticas.
- Possibilidade de ponderar as probabilidades de múltiplas hipóteses para classificar novas instâncias.

O classificador *Naive Bayes* é treinado a partir de um conjunto de exemplos de cada classe, descritos por um grupo de valores de atributos. Esse classificador assume que os valores de cada atributo são condicionalmente independentes para a relação com a classe, ou seja, dada uma classe Y, não é necessário calcular a probabilidade de ocorrência de cada combinação dos valores de X, mas apenas a probabilidade de cada X_i. Com base nas probabilidades obtidas no treino, a classificação de uma nova instância é feita atribuindo-lhe a classe mais provável, dados os valores de seus atributos (TAN; STEINBACH; KUMAR, 2006).

2.3.2 Aprendizado não supervisionado

No aprendizado não supervisionado o sistema de aprendizado não necessita de um conjunto de treinamento, pois utiliza apenas as características e atributos dos dados para classificá-los, não considerando rótulos previamente definidos. O resultado esperado nesse tipo de aprendizado é a formação de agrupamentos ou *clusters* de acordo com a semelhança entre as características dos dados de entrada (GENTLEMAN; CAREY, 2008). Por não necessitar de rótulos, esse tipo de aprendizado não requer nenhum esforço manual para classificar exemplos para treinamento. Segundo Aggarwal e Zhai (2012, p. 5, tradução nossa), "os dois principais métodos de aprendizado não supervisionado comumente usados no contexto de dados textuais são *clustering* e *topic modeling*"[7].

2.3.2.1 Clustering

A técnica de *clustering,* ou *cluster analysis*, busca agrupar entidades com base em seus atributos, sem considerar classes possivelmente já definidas para as entidades. O objetivo é criar grupos para cada classe, onde suas entidades possuem similaridades entre si e diferenças com entidades de outros grupos (KAROUSSI, 2012). Esses grupos permitem que novas entidades sejam analisadas e classificadas a partir dos grupos já definidos. Conforme isso ocorre, os grupos são aperfeiçoados, pois, novos atributos podem ser descobertos com a análise das novas entidades, refinando a definição do grupo e aumentando a capacidade de inferência do sistema.

O *clustering* não é uma técnica exclusiva do campo de aprendizagem de máquina, mas tem sido aplicado em diversas áreas de pesquisa como estatística, otimização e geometria computacional, análise genética, identificação de doenças na área da saúde, análise de comportamento do consumidor, reconhecimento de padrão de imagem e segurança (KAROUSSI, 2012; HAN; KAMBER; PEI, 2012).

[7] The two main un- supervised learning methods commonly used in the context of text data are clustering and topic modeling. (AGGARWAL E ZHAI, 2012, p. 5)

2.3.2.2 Topic Modeling

Topic model (modelo tópico) é um modelo probabilístico utilizado para encontrar tópicos de textos. Através da frequência em que ocorrem e da combinação de ocorrências de palavras, o modelo é capaz de identificar um conjunto de tópicos ou temas em uma grande coleção de documentos e as palavras utilizadas nestes tópicos. Se, por exemplo, A e B regularmente ocorrem juntos, pode-se assumir que A e B são palavras de um mesmo tópico. Da mesma forma, se A e B nunca ocorrem juntas, pode-se assumir que são palavras de tópicos diferentes.

Topic modeling é muito útil para mineração de opiniões, pois é capaz de identificar funcionalidades e classificação de sentimentos em documentos de revisão de um produto, levando em consideração as diferentes palavras usadas em revisões positivas e revisões negativas, *topic modeling* pode identificar tópicos positivos e tópicos negativos (KIM; GANESAN, 2011).

2.3.3 Aprendizado semi supervisionado

Comparado ao aprendizado supervisionado, que usa apenas exemplos rotulados, o aprendizado semi supervisionado pode ter maior precisão por considerar também exemplos não rotulados. Porém, isso depende da relevância do conjunto de exemplos não rotulados para o problema a ser tratado. Se tais exemplos não possuem informações úteis para a solução, eles podem desviar a inferência, prejudicando a qualidade da previsão.

Esse tipo de aprendizado traz maior vantagem em casos em que há baixo custo para obter os exemplos, mas há alto custo para obter os rótulos, seja custo de tempo, esforço ou dinheiro. Várias aplicações de aprendizado de máquina tem esse tipo de custo, como nas áreas de reconhecimento de fala, classificação de páginas web e resoluções de estruturas de proteínas.

Em contrapartida, para maior exatidão da previsão do aprendizado semi supervisionado, são necessários grandes quantidades de dados não rotulados, pois eles trazem menos informação do que dados rotulados. Isso implica na necessidade de algoritmos rápidos e eficientes (OLIVIER; SCHÖLKOPF; ZIEN, 2006).

2.3.3.1 Máquina de Vetores de Suporte (Support Vector Machines)

As Máquinas de Vetores de Suporte (SVMs, do inglês *Support Vector Machines*) tem como base a teoria de aprendizado estatístico, que utiliza uma função de classificação (classificador) para definir uma separação (fronteira de decisão) entre as diferentes classes durante o treinamento do sistema de aprendizagem. Com tal separação é possível aplicar o classificador a entradas desconhecidas e não classificadas, para inferência de suas classes (LORENA, 2007; KIM, 2003). Algumas versões de algoritmos de SVM seguem o modelo de aprendizado semi supervisionado.

2.4 MINERAÇÃO DE OPINIÕES (*OPINION MINING*)

Análise de sentimento ou mineração de opiniões, do inglês *sentiment analysis* e *opinion mining*, respectivamente, "[...] é o campo de estudo que analisa as opiniões, avaliações, atitudes e emoções das pessoas sobre entidades como produtos, serviços, organizações indivíduos, problemas, eventos, tópicos e seus atributos"[8] (LIU, 2012, p.7, tradução nossa).

Técnicas de mineração de opinião podem ser aplicada em textos de diversos tamanhos em diferentes níveis e com diferentes objetivos. Quanto aos níveis a análise pode ser realizada a nível de documento, de sentença e de entidade/aspecto. Algumas das aplicações da mineração de opinião são a extração e sumarização automática de opiniões, integração automática de opiniões de várias fontes, detecção automática de revisões falsas, correção de revisões mal classificada e monitoramento de entidades específicas (BECKER, 2014).

Segundo Liu (2012), de forma geral, uma opinião é constituída de dois elementos básicos. Um *alvo* que pode ser uma entidade ou um aspecto de uma entidade e um *sentimento* em relação a este alvo. Então, ao expressar uma opinião, seja sobre um produto, serviço, ou qualquer outro assunto, o interlocutor está se referindo a uma representação do objeto que está sendo avaliado. Alguns autores

[8][...] is the field of study that analyzes people's opinions, sentiments, evaluations, appraisals, attitudes, and emotions towards entities such as products, services, organizations, individuals, issues, events, topics, and their attributes (LIU, 2012, p.7)

(LIU, 2012; TSYTSARAU e PALPANAS, 2012; BECKER, 2014) utilizam a denominação *feature* para se referir de forma genérica ao alvo da opinião quando a análise é realizada à nível de entidade/aspecto.

Tsytsarau e Palpanas (2012) diz que diversas técnicas de análise de sentimentos podem ser divididas em três macro-tarefas. São elas: Identificação das features; classificação da polaridade da opinião expressa; e sumarização. Sendo está última a menos complexa, podendo, inclusive, não estar presente em alguns casos ou ser tratada apenas como uma tarefa de pós processamento.

2.4.1 Descoberta de Aspectos

O objetivo da primeira etapa no processo de mineração de opinião é a descoberta do alvo da opinião. Um alvo, ou simplesmente uma *entidade 'e'*, de acordo com Liu (2012), pode ser definido como um par *(T, W)*, onde *T* é uma hierarquia de componentes, subcomponentes, etc de *'e'* e *W* é um conjunto de atributos de *'e'*. Cada componente ou subcomponente também possui seu próprio conjunto de atributos. Um objeto como um automóvel pode ser um exemplo de entidade. Ele possui um conjunto de componentes, como motor, rodas, pedais, etc, e também um conjunto de atributos, entre eles espaço interno, número de portas, espaço do bagageiro. O motor por sua vez possui os atributos potência, número de válvulas, e assim por diante.

Como uma entidade pode ser dividida em um indefinido número de níveis e, tendo em vista a dificuldade de representação de tais níveis, bem como considerando a complexidade de PLN e que a maioria das aplicações não necessita de um alto grau de detalhamento, Liu (2012) sugere a adoção de apenas dois níveis de hierarquia, com o nodo raiz representando a entidade em si e os nodos folha representando os aspectos de tal entidade.

Liu (2012) define *opinião* de forma mais formal como uma quíntupla ($\mathbf{e}_i$, $\mathbf{a}_{ij}$, $\mathbf{s}_{ijkl}$, $\mathbf{h}_k$, $\mathbf{t}_l$), onde $\mathbf{e}_i$ representa uma entidade, $\mathbf{a}_{ij}$ é um elemento opcional que representa um aspecto de $\mathbf{e}_i$, $\mathbf{s}_{ijkl}$ é a polaridade em relação ao aspecto $\mathbf{a}_{ij}$, $\mathbf{h}_k$ é o autor da opinião e $\mathbf{t}_l$ é o momento do tempo em que a opinião foi expressa. Esta definição apresenta alguns atributos adicionais (e.g. autor e momento) que podem não ser necessários em alguns escopos de aplicação. Neste formato, geralmente, o aspecto

GERAL é utilizado para denotação da entidade como um todo. A utilização de uma definição deste tipo facilita a visualização da opinião como um dado estruturado e possibilita a manipulação dos dados de forma facilitada na etapa de sumarização.

No âmbito de revisão de produtos e serviços, geralmente, a opinião é expressa sobre um atributo de uma entidade, também denominado *feature*. Tendo em vista que assume-se que uma revisão trata de uma única entidade a principal dificuldade é determinar sob qual *feature* o autor está expressando opinião. Isto geralmente é feito buscando-se por palavras de sentimento ou classes de palavras candidatas a expressar sentimentos (BECKER, 2014).

A identificação de *features* por Hu e Liu (2004a) é feita utilizando-se uma busca por palavras pertencentes à classe morfológica substantivo. Tendo em vista que esta é a forma mais comum em que as *features* aparecem. Conforme citado por Becker (2014), alguns autores (QIU et al. 2011; LIU et.al. 2013) utilizam uma árvore de dependências sintáticas derivada a partir da sentença para identificar com maior precisão o alvo da opinião.

2.4.2 Classificação da Polaridade

A tarefa de descoberta das *features*/aspectos contidas em um documento é sucedida pela classificação de polaridade, que tem como objetivo identificar a orientação do sentimento que se refere a cada aspecto encontrado (KIM et al., 2011). Kim et al. (2011) descreve orientação de forma binária, podendo ser positiva ou negativa, porém Liu (2011) inclui a orientação neutra, que na prática é interpretada como ausência de opinião sobre o aspecto. Tsytsarau e Palpanas (2012) reforçam estas classes para classificação, mas também indicam a possibilidade de utilização de diversas classes (e.g. muito positivo, moderadamente positivo) ou ainda intervalos numéricos para representar intensidade. A tarefa de classificação de polaridade é aplicada sobre cada sentença que contém algum aspecto, identificando a polaridade de opinião da sentença como um todo e atribuindo tal polaridade à opinião direcionada ao aspecto (LIU, 2011).

As abordagens para classificação de polaridade, segundo Tsytsauro e Palpanas (2012), reforçados por Becker (2014), podem ser divididas em quatro. a) aprendizado de máquina, com o uso predominante de técnicas de classificação; b)

léxicas, com o uso de dicionários de sentimentos; c) estatísticas, que valem-se de técnicas estatísticas para avaliar a o-ocorrência de termos, e d) semânticas, que definem a polaridade de palavras em função da sua proximidade semântica com outra de polaridade conhecidas.

2.4.2.1 Abordagem de aprendizado de máquina

Segundo Tsytsauro e Palpanas (2012, p. 484, tradução nossa)

> A abordagem de aprendizado de máquina é uma solução sofisticada para o problema de classificação que pode ser geralmente descrita como um processo de duas partes: (1) aprender o modelo a partir de um conjunto de dados de treinamento (supervisionado, não supervisionado), e (2) classificar a base de dados desconhecidos utilizando o modelo treinado. [9]

Esta abordagem utiliza algumas das técnicas de aprendizagem de máquina citadas na seção 2.3 para, a partir de um conjunto de dados de treino, inferir um modelo de classificação que possa ser aplicado à novas entradas. Este modelo depende de dados de treino, que, em alguns casos, podem ser rotulados de forma manual, representando um alto custo para o desenvolvimento. Há também a possibilidade de utilizar as notas, que geralmente são associadas aos *reviews*, para derivar classes e associá-las aos mesmos, representando um facilitador na implementação.

2.4.2.2 Abordagem léxica

Para definir a orientação semântica de palavras de opinião com polaridade desconhecida são utilizados léxicos de sentimentos. Estes léxicos são dicionários contendo um grande conjunto de palavras ou expressões de sentimento com as suas respectivas polaridades (BECKER, 2014). Como regra geral, tais dicionários são dependentes do idioma abordado. Becker (2014) aponta como léxicos disponíveis para a língua portuguesa o OpLexicon (SOUZA et al. 2011) e o

[9] The Machine Learning Approach is a sophisticated solution to the classification problem that can be generally described as a two-step process: (1) learn the model from a corpus of training data (supervised, unsupervised), and (2) classify the unseen data based on the trained model. (TSYTSAURO E PALPANAS, 2012, p. 484)

SentiLex-PT (SILVA et al. 2012) sendo o primeiro para português do Brasil e o último para português de Portugal. Também é apontado o Linguistic Inquiry and Word Counts (LIWC) (TAUSCZIK e PENNEBAKER, 2007), este com versão para multi idiomas.

De acordo com Tsytsauro e Palpanas (2012) a abordagem léxica apresenta uma limitação ao tratar de palavras de sentimento de contextos específicos. Tendo em vista que o contexto pode alterar a polaridade da palavra, bem como que algumas destas palavras podem não estar contidas nos léxicos disponíveis.

2.4.3 Sumarização

Liu (2011, p. 467, tradução nossa) observa que "a maioria das aplicações de mineração de opinião precisam analisar opiniões de um grande número de detentores de opinião"[10]. Para permitir um uso eficiente dos resultados obtidos com as etapas de descoberta de *features* e classificação de polaridade, são necessárias formas organizadas de apresentar as informações mais relevantes em um resumo de fácil entendimento (KIM et al., 2011; NISHIKAWA, 2010). Esse é o objetivo da tarefa de sumarização dentro da mineração de opiniões.

Segundo Kim et al. (2011), o formato de sumarização mais adotado é o sumário estatístico, que apresenta o número de opiniões positivas e negativas para cada aspecto encontrado, facilitando a visualização e o entendimento de todo o resultado ou de um aspecto específico. Esse tipo de sumário pode ser apresentado também com o auxílio de gráficos, e os números de cada polaridade podem ser acompanhados das sentenças que contém tais opiniões.

[10] most opinion mining applications need to study opinions from a large number of opinion holders (LIU, 2011, p. 467)

3. TRABALHOS RELACIONADOS

3.1 TOKENIZAÇÃO

A etapa de tokenização é fundamental para todos as aplicações de PLN. E por isso os algoritmos utilizados devem levar em consideração as características da linguagem analisada bem como o objetivo da aplicação. Algumas regras podem ser empregadas para separar um texto em sentenças e em tokens. Tokenizadores baseados em espaços em branco utilizam espaços, tabulações e quebras de linha como separadores. Existem também tokenizadores *Treebank style*, que consideram como separadores a maioria dos caracteres de pontuação.

No contexto de mineração de opinião Cristopher Potts (2011, apud BECKER, 2014) propõem algumas heurísticas para tokenização. Entre elas o reconhecimento de emoticons, bem como marcações específicas da internet, xingamentos e siglas representando sentimentos. Neste âmbito os tokenizadores também podem ser utilizados para tratar formas de intensidade, como alongamento de palavras e utilização de letras maiúsculas, preservação de nomes próprios e também identificação de expressões idiomáticas.

Becker (2014) cita como ferramentas disponíveis para tokenização de textos em português do Brasil o NLTK[11], na linguagem Python, o Palavras[12], um software proprietário. E os sites LX_Center[13] e Linguateca[14].

3.2 STEMMING E LEMATIZAÇÃO

Como já mencionado na seção 2.1.1, algumas técnicas de normalização podem ser utilizadas com o objetivo de representar um grupo de palavras através de um único termo. Entre elas o *stemming* e a *lematização*.

O *stemming* é utilizado para redução de um termo ao seu radical através da remoção de desinências, afixos e vogais temáticas. Cada algoritmo possui um conjunto de regras que são dependentes do idioma tratado. Becker (2014) cita como

[11] www.nltk.org
[12] visl.sdu.dk
[13] lxcenter.di.fc.ul.pt
[14] www.linguateca.pt

exemplo de *stemmers* para o português do Brasil o RSLP[15] e o Snowball[16]. Apesar de ser bastante utilizada na área de recuperação de informação, no âmbito de análise de sentimentos, segundo Potts (2011, apud BECKER, 2014), tal técnica geralmente não gera bons resultados pois a eliminação dos afixos, a fim de se obter o radical, pode tornar impossível a diferenciação de polaridade de termos com o mesmo radical.

A lematização por sua vez é “o processo de agrupar diferentes inflexões e variantes de um termo para que possam ser analisadas como um único item, o lema” (BECKER, 2014, p. 134). Segundo Becker, uma das melhores opção de lematizador para o português do Brasil é o NLTK.

3.3 SUMARIZAÇÃO DE OPINIÕES BASEADA EM ASPECTOS

Segundo Kim et al. (2011, p. 6, tradução nossa), “O tipo de técnica de sumarização de opiniões mais comum é a sumarização de opiniões baseada em aspectos”[17]. Em geral, essas técnicas são aplicadas em 3 etapas, *aspect/feature identification*, *sentiment prediction*, e *summary generation*.

3.3.1 Aspect/feature identification

Essa etapa é usada para identificar tópicos notáveis no texto a ser sumarizado. Em alguns casos, porém, assume-se que os tópicos já são conhecidos, não sendo necessária essa etapa. Alguns trabalhos utilizam para essa etapa abordagens baseadas em duas diferentes áreas: Natural Language Processing (NLP) (HU e LIU, 2004a, 2004b; LU et al., 2009; POPESCUAND e ETZIONI, 2005) e *Data Mining* (ARCHAK et al., 2007; HU e LIU, 2004a, 2004b; POPESCUAND e ETZIONI, 2005).

Do contexto de NLP, Lu et al. (2009) utiliza *shallow parsing*, identificando aspectos em comentários curtos, onde o *parser* identifica opiniões como pares de termo principal (*head term*), que faz referência a um aspecto, e modificador, que

[15] http://www.inf.ufrgs.br/~viviane/rslp
[16] http://snowballstem.org/
[17] The most common type of opinion summarization technique is Aspect-based Opinion Summarization (KIM et al., 2011, p. 6)

expressa alguma opinião a respeito desse aspecto; Popescu e Etzioni (2005) faz uso do sistema KnowItAll (ETZIONI et al. 2004), um sistema de extração de informação baseado na web e independente de domínio, que extrai orações substantivas e encontra características de um determinado produto e de conceitos relacionados a ele.

De *Data Mining*, Hu e Liu (2004a, 2004b) utilizam regra de associação supervisionada, com o objetivo de inferir *feature words* a partir de segmentos de sentenças; Zhuang et al. (2006) usa expressões regulares para identificar se uma palavra em uma avaliação de filme coincide com uma das palavras da lista de *features*, que é formada pela combinação de todo o elenco de cada filme; Ku et al. (2006) utiliza a frequência das palavras, considerando como features as palavras mais frequentes a nível de parágrafo e a nível de documento; Archak et al. (2007) aborda a mineração de texto combinada a técnicas econométricas, decompondo o texto em segmentos que avaliam características individuais de um produto, estimando o peso e a pontuação que os consumidores atribuem a cada característica, e estimando o quanto cada avaliação afeta o retorno financeiro do produto.

3.3.2 Sentiment Prediction

Nessa segunda etapa de *opinion summarization* baseada em aspectos, o propósito é descobrir a orientação do sentimento (positivo ou negativo) relacionado ao aspecto/*feature*. Para essa etapa, alguns trabalhos utilizam métodos baseados em aprendizado (LU et al., 2009), onde a polaridade de um aspecto é estimada a partir da polaridade de frases relacionadas a ele; e outros que utilizam métodos baseados em *lexicons* (HU e LIU, 2004a; 2004b; KU et al., 2006; ZHUANG et al., 2006), onde a polaridade de uma sentença é definida pela quantidade de palavras positivas ou negativas que a formam, considerando listas pré-definidas de palavras positivas e negativas.

3.3.3 Summary generation

O objetivo da etapa de *summary generation* é gerar e apresentar uma opinião

final resumida em um formato eficaz e fácil de interpretar. Os trabalhos na área utilizam diferentes formatos de resumos. Hu e Liu (2004a, 2004b, 2006) e Zhuang et al. (2006) introduzem *statistical summary*, que é o formato mais comumente adotado (KIM et al., 2011) e que consiste em apresentar o número de opiniões positivas e negativas de cada aspecto e as sentenças que expressam tais opiniões; alguns trabalhos (KU et al., 2006; LU et al., 2009; MEI et al., 2007; POPESCUAND e ETZIONI, 2005; TITOV e MCDONALD, 2008) utilizam *text selection* para apresentar pequenos trechos de texto representando as opiniões mais fortes sobre cada aspecto, o que fornece mais detalhes sobre as opiniões resumidas; Lu et al. (2009) propõe o tipo *aggregated ratings*, que une *statistical summary* e *text selection*, apresentando uma média dos sentimentos estimados nas frases sobre cada aspecto, junto com uma frase que represente o sentimento geral; Ku et al. (2006) e Mei et al. (2007) mostram as tendências das opiniões e as mudanças que podem ter ocorrido no decorrer do tempo, usando o tipo *summary with a Timeline*.

3.4 SUMARIZAÇÃO DE OPINIÕES NÃO BASEADA EM ASPECTOS

Kim et al. (2011) considera que essa classe engloba as abordagens que não se enquadram no formato baseado em aspectos, pois sugerem formatos diferentes para a sumarização de opiniões. Alguns trabalhos desse tipo abordam os seguintes tipos de sumarização: *basic sentiment summarization* (KIM et al., 2011), que conta e apresenta o número de opiniões positivas e negativas sobre cada sentença ou documento; *text summarization* (BALAHUR e MONTOYO, 2008; GANESAN et al., 2010; KIM e ZHAI, 2009; LU e ZHAI, 2008), que em suas abordagens envolve a integração entre documentos de diferentes níveis de especialização, a representação de sentenças de opiniões contrastantes, a representação gráfica das palavras e suas ligações, e a sumarização mapeada entre diferentes idiomas; *visualization* (CHEN et al., 2006; MISHNE et al., 2006), que foca em apresentar o resumo de opiniões de diferentes formas, para dar mais intuição aos leitores e aumentar a legibilidade; e *entity-based summarization* (STOYANOV e CARDIE, 2006b, 2006a, 2008), que mostra as entidades do texto e suas relações de fonte e alvo das opiniões.

4. CLASSIFICAÇÃO E SUMARIZAÇÃO DE COMENTÁRIOS

Para alcançar os objetivos deste trabalho, foi proposta uma ferramenta de classificação e sumarização de comentários textuais sobre produtos e suas características, aqui denominadas *domain features*. Para exemplificar e avaliar a proposta, implementou-se um software como protótipo da ferramenta, o qual recebe como entrada um conjunto de comentários, escritos em português do Brasil, sobre determinado produto. Sobre esses textos, o software desempenha várias etapas de tratamento através de técnicas de PLN, classifica cada sentença quanto à sua polaridade, utilizando um modelo de classificação treinado de forma supervisionada, e sumariza as polaridades relacionadas a cada *domain feature*.

Para melhor análise dos resultados, durante o desenvolvimento deste trabalho, considerou-se comentários textuais sobre produtos da categoria de celulares, assumindo que as características pertinentes ao processamento e resultado da ferramenta são apenas aquelas que descrevem um aparelho de telefone celular e sua qualidade.

O conjunto de comentários foi obtido de um site de comparação de preços, através de um *web crawler* (sistema de navegação automática e metódica pela internet, buscando informações pré-definidas em sua configuração), e armazenado em uma base de dados local para uso durante as etapas de desenvolvimento e testes.

Na etapa de classificação dos comentários, foram utilizados os algoritmos MaxEnt GIS (*Generalized Iterative Scaling*), MaxEnt IIS (*Improved Iterative Scaling*), Naive Bayes e Decision Tree, através da biblioteca NLTK, para comparação dos resultados obtidos através de cada algoritmo.

Como estrutura geral da ferramenta desenvolvida, foram definidos um fluxo para o processo de treinamento da mesma, representado pela Figura 1, e outro para o seu uso no processo de classificação e sumarização, exemplificado na Figura 2.

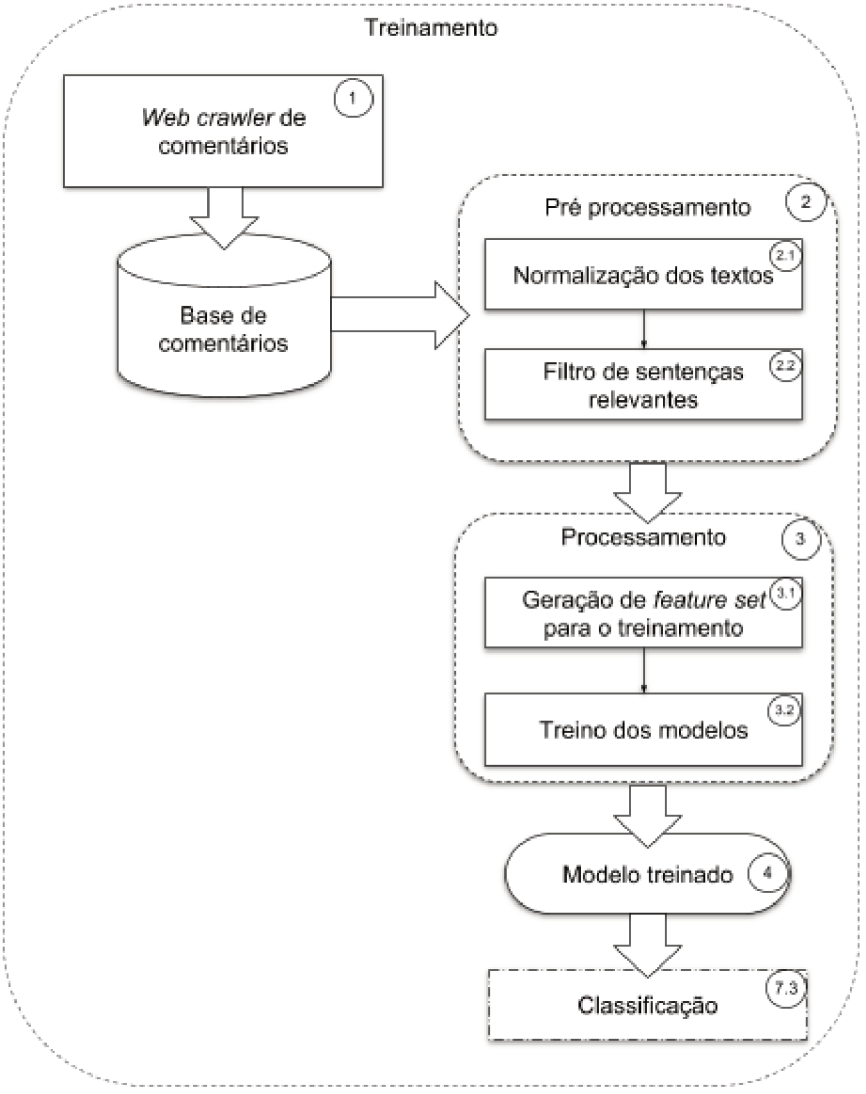

Figura 1 - Fluxo básico de treinamento do classificador

O processo de treinamento inicia na etapa 1, onde um *web crawler* é utilizado para a obtenção de um conjunto de comentários e notas. Estes dados são persistidos em um banco de dados para utilização futura. Na etapa 2 é realizado o pré-processamento dos comentários. Primeiramente é realizada a normalização dos textos (etapa 2.1) incluindo a remoção de links, letras e pontuações repetidas, etc. Os comentários normalizados são separados em sentenças e então são filtrados (etapa 2.2) a fim de evitar o processamento de textos irrelevantes. O conjunto de sentenças que não foram removidas na etapa 2 são enviados para a etapa 3, processamento. Nesta etapa são gerados os *feature sets* (etapa 3.1) contendo as características que serão utilizadas para o treinamento do modelo (etapa 3.2). Na etapa 4 o processo de treinamento gera como saída um modelo de classificação.

Este modelo será utilizado posteriormente na etapa 7.3 do processo de classificação e sumarização.

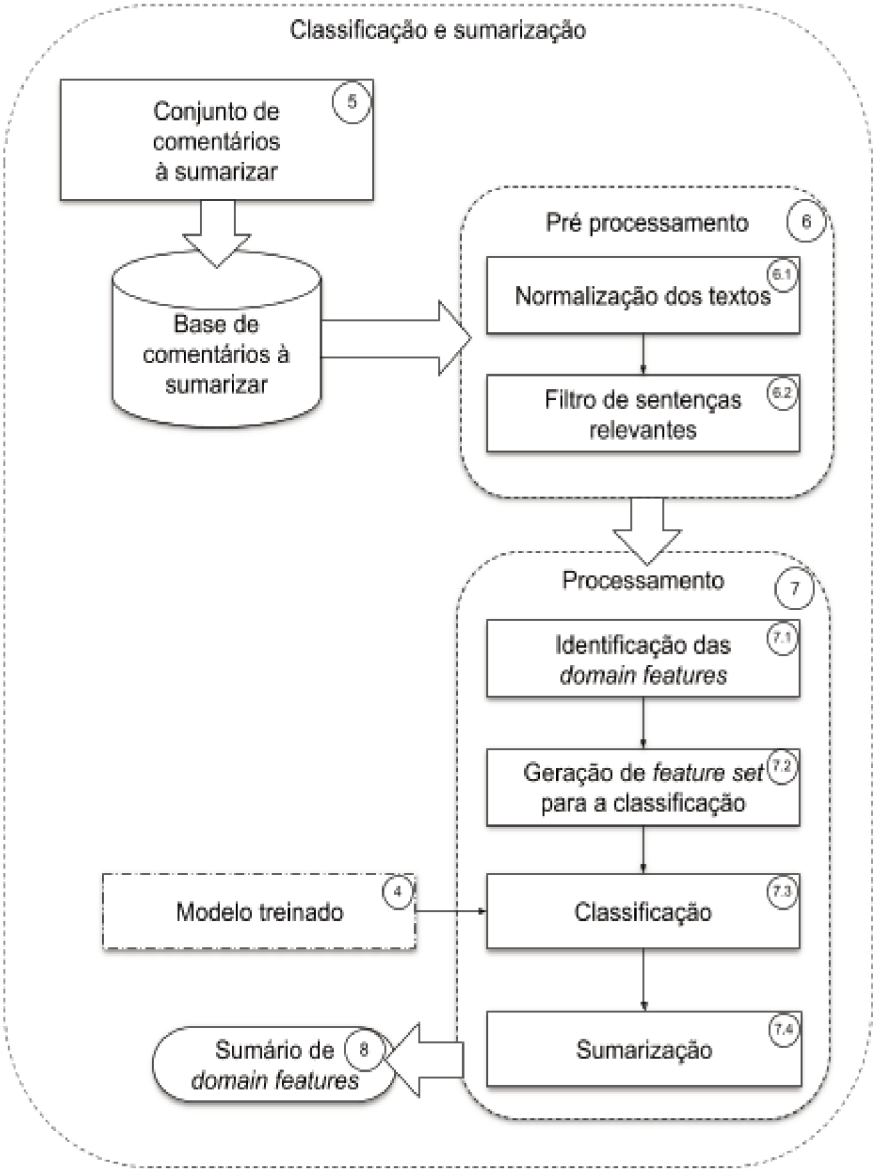

Figura 2: Fluxo básico da classificação e geração do sumário

O processo de classificação e sumarização inicia na etapa 5, com o conjunto de comentários sobre o qual desejamos realizar a classificação e geração de um sumário. Estes comentários são persistidos em uma base de dados para possibilitar possíveis análises futuras. Na etapa de pré-processamento (etapa 6) são realizados os mesmos tratamentos da etapa 2 do modelo de treinamento. O conjunto de sentenças que não foi removido na etapa de filtro (etapa 6.2) é processado na etapa 7. Primeiramente é realizada a identificação das *domain features* presentes na

sentença (etapa 7.1). Em seguida são gerados os *feature sets* (etapa 7.2) contendo as características relevantes para a classificação, que é realizada na etapa 7.3 utilizando o modelo resultante da etapa 4 do processo de treinamento. Na etapa 7.4 é realizada a sumarização do resultado da classificação associada a cada *domain feature*. Na etapa 8 é realizada a geração do sumário de *domain features*. Esta é a saída final do processo.

Nas seções seguintes são detalhadas as etapas desses processos, no modelo proposto para a sumarização de comentários e no protótipo desenvolvido.

4.1 MODELO PARA CLASSIFICAÇÃO E SUMARIZAÇÃO DE COMENTÁRIOS

Baseando-se nas tecnologias e ferramentas estudadas durante esse trabalho, foi elaborado um modelo de ferramenta capaz de analisar um conjunto de comentários sobre produtos, escritos em português, e listar de forma sumarizada os principais aspectos positivos e negativos relacionados ao objeto alvo dos comentários. O modelo proposto, que engloba os processos apresentados nas figuras 1 e 2, é detalhado nas próximas seções.

4.1.1 Entradas

Além do conjunto de comentários a serem sumarizados, o modelo depende de entradas que auxiliam nas etapas intermediárias. Essa seção descreve as entradas necessárias para o funcionamento do modelo de sumarização.

4.1.1.1 Comentários

Os comentários a serem processados e sumarizados, são encontrados comumente em sites de lojas online, comparação de preços ou avaliação de produtos. Para obtenção de comentários dessas fontes, propõe-se o uso de *web crawlers* (etapa 1 da Figura 1) configurados especificamente para cada site, de forma que seja possível obter os conteúdos dos comentários, além do nome, modelo e categoria do produto aos quais o conjunto de comentários se refere. As informações que identificam o produto em si são necessárias para fins de separação

contextual, pois as características (*domain features*) e qualificações de determinado produto/categoria podem representar uma polaridade inversa para outro produto/categoria. Por exemplo: em “o óculos é enorme” o adjetivo “enorme” tem uma conotação negativa, enquanto em “o porta malas é enorme” o mesmo adjetivo tem uma conotação positiva. Sendo assim, o modelo proposto assume que o conjunto de comentários sendo processado é sobre uma única categoria de produto.

Para treinar o classificador de polaridade (Figura 1), se faz necessária também uma nota numérica relacionada a cada comentário, a qual indicará a polaridade do mesmo. Portanto, os conjuntos de dados para treino e teste do classificador devem ser obtidos de fontes que apresentem uma nota numérica atribuída pelo autor do comentário.

4.1.1.2 *Domain Features*

Considerando que as características que descrevem um produto variam conforme a categoria a que este pertence, o modelo proposto depende da lista de *domain features* a serem consideradas no processamento dos comentários. Essa lista pode ser pré definida manualmente, elencando as características mais relevantes para a categoria de produto, ou podem ser obtidas de forma dinâmica, através de busca em fontes externas (bases de conhecimento, bibliotecas de ontologias etc.). As *domain features* são importantes para o filtro realizado no pré-processamento (etapas 2.2 da Figura 1, e 6.2 da Figura 2), onde a relevância das sentenças é avaliada, e para a sumarização (etapa 7.4 da Figura 2), que gera o resultado final do modelo.

4.1.1.3 Palavras polarizadas

O modelo depende também de mais duas listas, uma contendo as palavras consideradas como expressões de polaridade positiva; e outra, as palavras de polaridade negativa. De forma similar à obtenção da lista de *domain features*, as listas de palavras polarizadas podem ser obtidas de forma fixa, a partir de uma lista pré definida manualmente; ou de forma dinâmica, a partir da busca em fontes externas. Essas listas são importantes para a criação dos *feature sets* de treino do

classificador (etapa 3.1 da Figura 1), e os da classificação de polaridades (etapa 7.2 da Figura 2).

4.1.2 Pré-processamento

O processamento dos textos obtidos das fontes citadas na seção anterior é bastante afetado pela irregularidade gramatical de alguns comentários. Outro fator que influencia no resultado do processamento dos textos é que alguns comentários não expressam sentimentos sobre o produto em si ou suas características, sendo irrelevantes para o modelo proposto.

Para melhorar a qualidade dos textos e evitar o consumo desnecessário de tempo no processamento de comentários irrelevantes, os comentários são submetidos a um pré-processamento, que consiste nas etapas abaixo.

- Normalização dos dados (etapas 2.1 da Figura 1, e 6.1 da Figura 2): alguns detalhes do texto são alterados para garantir uma mínima padronização dos dados sendo processados. Nessa etapa ocorrem as tarefas de:
    a. Conversão de todas as letras para minúsculas, pois, dependendo da implementação, as etapas seguintes podem ser sensíveis à capitalização das palavras.
    b. Remoção de URLs, espaços em excesso e pontuação duplicada, por não serem relevantes para análise de polaridade.
    c. Conversão de quebras de linha para ponto final, para melhor resultado na separação das sentenças.
    d. Correção de erros gramaticais comuns, eliminando diferenças que possam descaracterizar as palavras do vocabulário e afetar a análise sintática.
- Divisão dos comentários em sentenças: etapa necessária para permitir a análise de polaridade a nível de sentença, tendo em vista que o modelo atribui a polaridade da sentença a todas as *domain features* contidas nela.
- Filtro de sentenças relevantes (etapas 2.2 da Figura 1, e 6.2 da Figura 2): essa etapa descarta as sentenças que não possuem termos que

indiquem sentimento positivo ou negativo, ou que não fazem referência a nenhuma das *domain features* predefinidas, pois, no segundo caso, ainda que haja expressão de sentimento, o modelo não é capaz de identificar a característica do produto à qual o sentimento se relaciona.

4.1.3 Processamento

É nesta etapa (3 na Figura 1, e 7 na Figura 2) que as sentenças que não foram descartadas pelo pré-processamento serão utilizadas para o treinamento de um modelo de classificação o qual, posteriormente, irá classificar as novas entradas em uma das duas classes propostas (positivo e negativo).

Para encontrar as propriedades de cada sentença a serem consideradas na classificação, são utilizadas técnicas de PLN para reconhecer os elementos de significado do texto. Neste modelo são utilizadas técnicas da categoria de análise morfológica nas etapas de:

- Tokenização: durante esta etapa as sentenças são subdivididas em tokens. Os tokens são os componentes da sentença, ou seja, as palavras e os caracteres de pontuação que a compõem.
- POS tag: nesta etapa os tokens gerados na etapa anterior são rotulados com a sua respectiva classe gramatical dentro da sentença. São estes rótulos que permitirão ao software diferenciar um adjetivo de um verbo, por exemplo.

Com os tokens rotulados pela aplicação de PLN, a cada sentença é associado um conjunto de propriedades (etapas 3.2 na Figura 1, e 7.2 na Figura 2), aqui denominado *feature set*, que será utilizado na classificação. Neste modelo, as propriedades que compõe o *feature set* de cada sentença, ou seja, que são consideradas determinantes para identificação da polaridade de cada sentença, são:

- Contém polaridade positiva: indica se tem ao menos uma palavra da lista de palavras positivas.
- Contém polaridade negativa: indica se tem ao menos uma palavra da lista de palavras negativas.
- Contém negação: indica se contém algum termo de negação que possa inverter a polaridade da sentença.

Esse modelo considera o uso de algoritmos de classificação por aprendizado supervisionado, sendo necessário o conhecimento da classe de cada sentença para o treino do classificador. Conforme apresentado anteriormente, no conjunto de comentários obtidos, os que se destinam ao treino possuem uma nota atribuída pelo avaliador. Durante as etapas de pré-processamento (2 da Figura 1, e 6 da Figura 2) e PLN, a nota do comentário é replicada para as sentenças que o compõem e a relação entre a nota e a sentença é mantida até a etapa de treino. No treino (etapa 3.2 da Figura 1), cada nota é discretizada para uma das classes, a classe é associada ao *feature set* correspondente, e esses pares (classe - *feature set*) são submetidos ao algoritmo de treino.

Após o treino, o modelo de classificação criado (item 4 da Figura 1) é usado para classificar novos comentários (etapa 7.3 da Figura 2), os quais não precisam de nota atribuída. Como resultado da classificação desse novo conjunto de comentários, tem-se cada sentença e sua respectiva polaridade, que são as entradas para a etapa de sumarização (7.4 da Figura 2).

4.1.4 Saídas

O modelo proposto gera como saída um documento composto de um sumário de *domain features* do produto analisado. As tarefas necessárias para a geração deste documento são descritas na seção 4.1.4.1 a seguir.

4.1.4.1 Sumarização

A etapa de sumarização (7.4 da Figura 2) é a responsável por gerar a saída da ferramenta, um sumário estatístico, de forma que sejam apresentadas as características mais comentadas sobre o produto, bem como a contagem de comentários positivos e negativos sobre cada uma dessas características.

A partir do resultado da classificação, a sumarização é feita em três etapas:

- identificação das *domain features* que constam em cada sentença;
- identificação dos comentários responsáveis pela polaridade de cada *domain feature*;

- contagem dos comentários, agrupados por *domain feature* e polaridade.

A contagem é feita a nível de comentário, não a nível de sentença. Contar a nível de sentença, ou seja, contar as sentenças que referenciam uma característica, poderia distorcer o resultado, tendendo à opinião de avaliadores que foram repetitivos em seus comentários e escreveram várias sentenças sobre a mesma característica. Na contagem a nível de comentário não há essa distorção, pois o comentário inteiro é considerado como uma única manifestação de opinião para cada característica encontrada nele. Supondo, por exemplo, que um único comentário contenha 5 sentenças positivas sobre uma mesma característica do produto, e todas as 5 sentenças foram processadas e classificadas. A nível de sentença, a contagem seria 5 para a polaridade positiva da característica comentada, mesmo sendo a opinião de uma única pessoa. Enquanto a nível de comentário, como no modelo proposto, a contagem seria 1.

Feita a contagem, a sumarização é apresentada como saída da ferramenta (item 8 da Figura 2), elencando cada *domain feature* encontrada no conjunto de comentários, juntamente com o número de comentários que a avaliam de forma positiva, o número de comentários que a avaliam de forma negativa e o número total de comentários sobre a *domain feature*. A lista apresentada é ordenada pelo total de comentários de cada item, em ordem decrescente, considerando que as características mais comentadas provavelmente são as mais relevantes para descrever o produto e auxiliar na avaliação.

4.2 ESTUDO DE CASO

Esta seção descreve o contexto específico ao qual o desenvolvimento e avaliação do protótipo foram direcionados. O foco em um contexto específico tem o objetivo de alinhar o esforço do trabalho com o tempo hábil disponível, bem como, validar a eficácia do modelo proposto em um contexto com menor variação nos valores de entrada.

4.2.1 Categoria de Produto

Considerando que no contexto de comentários sobre produtos as características comentadas variam conforme a categoria do produto, e que o modelo proposto considera que, a cada novo conjunto de dados submetido, todos os comentários são sobre produtos da mesma categoria, o desenvolvimento do protótipo considerou apenas a categoria de celulares.

4.2.1.1 Comentários

Na obtenção dos comentários sobre celulares, foi utilizado como fonte o site Buscapé[18], que é uma ferramenta gratuita de comparação de preços. Atualmente, o site permite o acesso a mais de 25 milhões de produtos em diversas lojas online, centralizando informações de preço atual, histórico de preços, características de produtos e comentários sobre os produtos, submetidos pelos consumidores. Em comparação com outros sites analisados durante a pesquisa por possíveis fontes de dados, o Buscapé se destacou pela grande quantidade de modelos de celulares disponíveis, a grande quantidade de comentários sobre os mesmos, e por possuir uma nota numérica (apresentada no site como quantidade de estrelas) associada a cada comentário, necessária para o processo de treino no modelo proposto, conforme detalhado na seção 4.1.1.1.

No site Buscapé, foram obtidos pouco mais de 12 mil comentários sobre uma grande variedade de modelos de celulares. Contendo informações como URL da página de origem, URL direta para o comentário, título, conteúdo, data, pontuação, total de avaliações como útil, total de avaliações como inútil e se o usuário recomenda o produto. Tendo em vista a forma de obtenção dos dados e a fim de garantir a unicidade dos mesmo, sobre estes aplicaram-se alguns filtros iniciais. Primeiramente foram mantidos apenas os comentários que possuíam valores para os atributos URL direta, conteúdo e pontuação. Também foram adicionadas regras para garantir a unicidade da URL direta e da combinação de valores de título,

[18] https://www.buscape.com.br

conteúdo, data e pontuação. Resultando em 10272 comentários únicos que foram armazenados em um banco de dados.

Antes do uso do Buscapé, o Twitter foi analisado como fonte para obtenção dos comentários, assim como é feito em Salas-Zárate et al. (2017) e outros trabalhos da área, porém notou-se que haviam poucos dados com as características necessárias para o contexto sendo considerado. Foram aplicados filtros na busca dos *tweets* para obter apenas os que fizessem referência a alguma característica de celular e que contivessem ao menos um *emoticon* (sequência de caracteres que traduz um estado emotivo), o qual seria utilizado como rótulo para o treinamento do classificador. Com a aplicação dos filtros, pouco mais de 150 *tweets* foram obtidos, não sendo suficientes para o treinamento e teste de um modelo de classificação.

4.2.1.1 *Domain Features*

Como *domain features* para a categoria, foram selecionadas manualmente as características mais comuns em descrições de aparelhos celular, considerando anúncios em lojas virtuais, como Submarino[19], Americanas[20] e ShopTime[21]. Foram adicionadas também algumas características indiretamente relacionadas ao produto, mas que também influenciam a escolha no momento da compra, como assistência técnica, atendimento e garantia. As *domain features* selecionadas são listadas no Quadro 1.

Quadro 1 - Lista de *domain features* selecionadas

3g	garantia	tipo de tela	processamento
4g	gps	tv	dois chips
armazenamento	marca	wi-fi	dual chip
assistência técnica	memória ram	amoled	aparelho
atendimento	microfone	ips	celular
áudio	modelo	lcd	produto
banda	multichip	oled	smartphone
bateria	nfc	display	telefone
bússola	peso	visor	câmera frontal

[19] https://www.submarino.com.br
[20] https://www.americanas.com.br
[21] https://www.shoptime.com.br

câmera	processador	tamanho do display	câmera traseira
carregador	recursos	altura	autonomia
chip	resolução	comprimento	tipo de bateria
conexão	sistema operacional	largura	som
cor	tamanho	android	cartão de memória
dispositivo	tamanho da tela	ios	expansível
embalagem	tela	s.o.	memória interna

Algumas das *domain features* foram agrupadas em *features* principais, pois referenciam uma mesma característica do produto, diferenciando-se apenas pela forma de escrita ou idioma em que se encontram nas fontes de dados utilizadas. Esse agrupamento é, principalmente, para que o resultado final não apresente separadamente as características redundantes, mas agrupe seus sumários, somando as contagens de polaridades e apresentando como uma única *domain feature*, identificada pelo nome da feature principal. As features agrupadas são apresentadas no Quadro 2.

Quadro 2: *Domain features* agrupadas por *feature* principal

Feature principal	**Domain feature**
armazenamento	cartão de memória
	expansível
	memória interna
bateria	autonomia
	tipo de bateria
câmera	câmera frontal
	câmera traseira
dispositivo	aparelho
	celular
	produto
	smartphone
	telefone
multichip	dois chips
	dual chip
processador	processamento
sistema operacional	android
	ios
tamanho	altura
	comprimento
	largura
tamanho da tela	tamanho do display
tela	display
	visor
tipo de tela	amoled
	ips
	lcd
	oled
áudio	som

4.2.1.2 Palavras polarizadas

Para obtenção da lista de palavras polarizadas, que é uma das entradas previstas no modelo de classificação deste trabalho, optou-se pela definição manual da mesma. A escolha das palavras foi baseada em conhecimento de senso comum, selecionando palavras comumente utilizadas para descrever sentimento sobre um objeto sendo avaliado, e seus sinônimos, diretos e indiretos, mais relevantes para o contexto de celulares. O Quadro 3 apresenta as palavras selecionadas.

Quadro 3: Palavras polarizadas selecionadas

Palavras positivas		
acessível	extraordinário	positivo
adorei	feliz	prático
amei	forte	rápida
boa	fácil	rápido
boas	gostei	satisfeito
bom	legal	superior
bonito	magnífico	surpreendente
contente	maravilhosa	ótima
durável	maravilhoso	ótimo
eficiente	melhor	útil
excelente	perfeito	

Palavras negativas		
desagradável	incorreto	miserável
desprezível	inferior	má
detestável	insatisfatório	negativo
errado	insatisfeito	ordinário
fajuto	insuportável	pior
feio	inúteis	podre
fraca	inútil	péssima
fraco	lamentável	péssimo
horrível	lento	ruim
imprestável	maldito	terrível

4.2.2 Classificação

O modelo proposto considera o uso de técnicas de classificação por aprendizado supervisionado. Essas técnicas são implementados por diversas ferramentas de PLN, porém, no escopo deste trabalho decidiu-se por utilizar a *toolkit* NLTK. Ele foi destacado por Becker (2014) dentre os *toolkits* disponíveis para tratamento da língua portuguesa, e é amplamente utilizado para propósitos acadêmicos e/ou comerciais. Além de sua popularidade, uma característica de destaque do NLTK é a facilidade de uso, que une a facilidade da linguagem Python, principalmente no contexto de PLN; a completude e clareza da documentação de

cada ferramenta disponível no *toolkit*; e a grande comunidade de usuários disponível na internet através de fóruns e artigos sobre a biblioteca, suas aplicações e resoluções de possíveis problemas.

Há diversos algoritmos disponíveis para realizar aprendizagem supervisionada, como KNN, redes neurais, Naive Bayes, Decision trees, Maximum Entropy (MaxEnt) entre outros. O *toolkit* NLTK possui a implementação dos algoritmos de classificação MaxEnt GIS (*Generalized Iterative Scaling*), MaxEnt IIS (*Improved Iterative Scaling*), Naive Bayes e Decision Tree. Ele possui também uma interface com a biblioteca scikit-learn[22], a qual implementa outros algoritmos de classificação.

Não havendo conhecimento prévio sobre qual dos algoritmos disponíveis na biblioteca NLTK seria o mais adequado para o modelo proposto, optou-se por abordar a classificação com todos os 4 algoritmos disponíveis, para posterior comparação dos resultados. O uso dos classificadores implementados pela biblioteca scikit-learn exigiria esforço extra para configurar algumas dependências dos mesmos e incluí-los no fluxo de classificação. Em razão disso, e de a comparação dos algoritmos não ser o objetivo principal do trabalho, os classificadores da biblioteca scikit-learn não foram utilizados.

Os mesmos subconjuntos de treino e teste foram utilizados para cada classificador. Na definição de tamanho desses subconjuntos, priorizou-se o treino, buscando submeter um maior número de exemplos na etapa de aprendizagem, para melhorar a acurácia do classificador. Além disso, o conjunto de dados obtido possui uma grande quantidade de exemplos (10272 comentários). Portanto, o conjunto de comentários foi dividido em 90% (9245) para treino e 10% (1027) para teste.

4.3 PROTÓTIPO

O modelo proposto foi utilizado como base para o desenvolvimento de um protótipo, cuja entrada é um conjunto de comentários textuais, escritos em português do Brasil, sobre aparelhos celulares e a saída é um documento, em formato de sumário, listando as *domain feature* com o total de comentários que classificaram tal

[22] http://scikit-learn.org

característica de forma positiva e negativa, bem como o somatório de ambas as classificações. Este somatório é utilizado para ordenar o sumário em ordem decrescente, tendo em vista que as características mais comentadas são as de maior destaque no produto analisado, e, possivelmente irão interessar para um maior número de usuários.

Um exemplo esquemático do protótipo desenvolvido para o treinamento e a classificação e sumarização dos comentários pode ser visto nas figuras 3 e 4, respectivamente. As etapas desenvolvidas, bem como as decisões de projeto tomadas durante o desenvolvimento são explicadas na sequência.

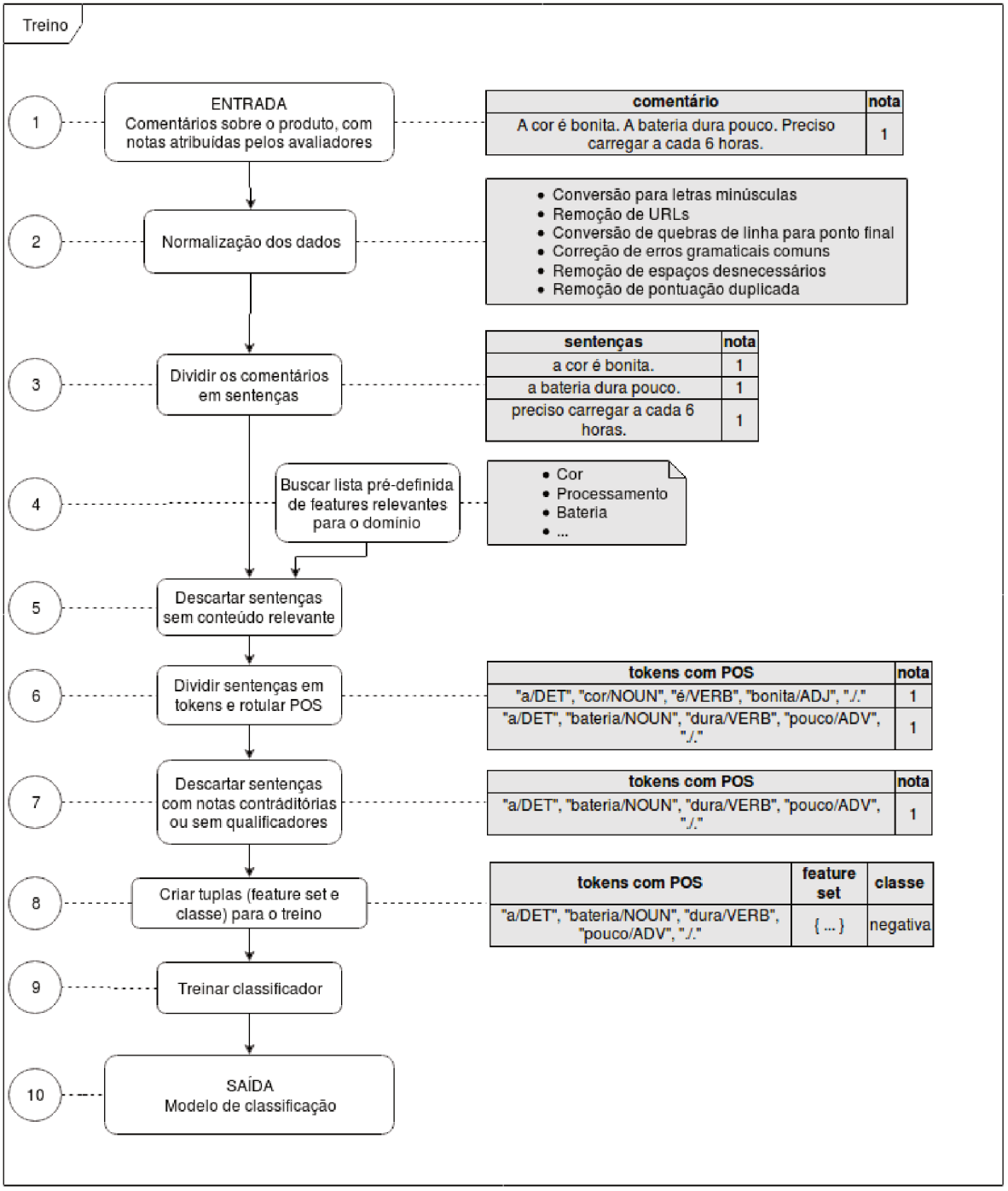

Figura 3: Esquema de treinamento do classificador

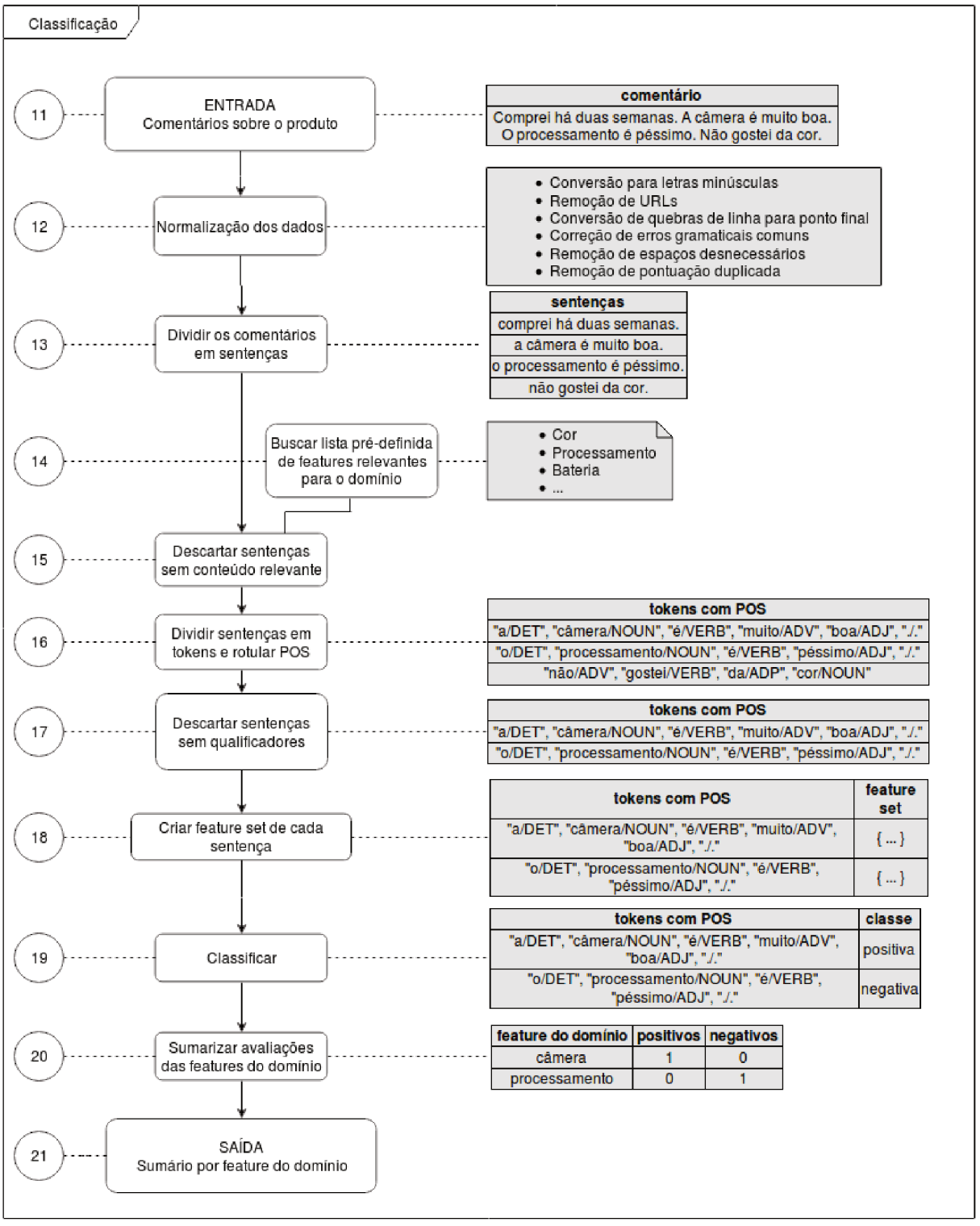

Figura 4: Esquema de classificação e sumarização

4.3.1 Entradas

O protótipo desenvolvido utiliza como entrada principal dois formatos de dados distintos. Um para o processo de treinamento e outro para o processo de classificação e sumarização. Para o treinamento do classificador, etapa 1 da figura 3, espera-se um conjunto de tuplas compostas de comentários textuais, escritos em português do Brasil, e uma nota numérica, de 1 a 5. Para a classificação e sumarização, etapa 11 da figura 4, o protótipo necessita apenas de um conjunto de comentários escritos em português do Brasil. Além destes são necessários dados para a identificação das *domain features*, etapas 4 e 14 das figuras 3 e 4, respectivamente, e dados para a identificação das palavras de sentimentos, que serão utilizadas em uma das etapas de filtragem, etapa 7 da figura 3, e também para geração dos *feature sets*, etapa 8 da figura 3. A seguir são explicados como cada conjunto de dados foi obtido.

4.3.1.1 Comentários

Para a criação de um modelo de classificação com alta precisão, é necessário um conjunto de dados de treinamento que engloba o maior número de cenários possíveis. Como a obtenção de um grande conjunto de dados de forma manual é inviável, foi utilizado um *web crawler* para realizar o *scraping* (extração de dados) de comentários de usuários sobre aparelhos celular. O site Buscapé foi escolhido entre as diversas opções disponíveis por agrupar informações provenientes de um grande conjunto de outros sites. Para realizar esta tarefa um *web crawler* preliminar foi implementado utilizando a linguagem de programação *python*. No entanto esta abordagem não foi eficiente, pois o site Buscapé utiliza um mecanismo de segurança que rejeita requisições realizadas em um curto intervalo de tempo a partir do mesmo IP. Buscando por opções de *web crawlers* disponíveis, que se adaptassem a esta característica, optou-se pela utilização da ferramenta Apify[23], que permite a implementação de um script para *scraping* utilizando a linguagem Javascript e utiliza um conjunto de IPs dinâmicos para realizar o acesso às URLs encontradas. A

[23] https://www.apify.com/

ferramenta disponibiliza uma faixa de utilização a nível gratuíto, permitindo analisar até 5000 páginas por mês. Este nível foi suficiente para a obtenção de um conjunto de 10272 comentários, os quais foram armazenados em um banco de dados e utilizados para o treinamento e avaliação do protótipo desenvolvido.

4.3.1.2 *Domain Features*

Para a obtenção das domain features, etapas 4 e 14 das figura 3 e 4, respectivamente, buscou-se uma ferramenta que permitisse a identificação das características de produtos de forma automática. Tendo em vista que esta abordagem permitiria a fácil extensão do protótipo para outros tipos de produtos. Foram realizados alguns testes manuais com a ferramenta DBPedia Spotlight, apresentados no APÊNDICE A, mas os resultados não foram os esperados. Considerando o escopo de aparelhos celulares, a ferramenta não se mostrou preparada para identificar as características do mesmo e por isso foi descartada. Em seguida foi utilizada a mesma abordagem escolhida para a obtenção dos comentários. Novamente a ferramenta Apify foi utilizada em conjunto de scripts de *scraping* para a captura dos dados. No entanto, após execuções preliminares percebeu-se que o trabalho despendido para a escrita dos scripts não era vantajoso, tendo em vista a necessidade de escrever um novo script para cada site, cujos dados não possuíam grande variação. Desta forma optou-se pela captura manual das características mais relevantes e suas variações, como descrito na seção 4.2.1.

4.3.1.3 Palavras polarizadas

As palavras polarizadas (palavras que denotam sentimento positivo ou negativo) foram obtidas através de um conjunto inicial baseado em conhecimento de senso comum (palavras como bom, ótimo, ruim, péssimo) e de busca por sinônimos de forma manual e recursiva em dicionários online a partir deste conjunto inicial. As palavras encontradas foram analisadas pelos autores para então serem incluídas no conjunto final ou descartadas. O conjunto de palavras selecionadas e utilizadas neste trabalho é apresentado no Quadro 3.

A realização da busca de forma automática foi considerada uma opção. Para tal, verificou-se a possibilidade de utilização de uma das ferramentas encontradas durante o estudo do referencial teórico, o OpLexicon e o SentiLex-PT. Enquanto a primeira se mostrava mais interessante por tratar de português do Brasil, a segunda apresentava, no momento da escrita deste trabalho, uma documentação mais extensa. Contudo, optou-se por manter a abordagem manual devido a complexidade para implementação. A utilização de busca automática está prevista para trabalhos futuros.

4.3.2 Pré-processamento

A etapa de pré-processamento engloba as etapas de normalização do texto para o processamento, exibida como item 2 da figura 3 e item 12 da figura 4, responsável por transformar o texto para um formato padrão; divisão dos comentários em sentenças, representada pelo item 3 da figura 3 e item 13 da figura 4, a fim de permitir análises a nível de sentença; e uma pré filtragem dos comentários, apresentada como item 5 na figura 3 e item 15 na figura 4, com objetivo de evitar o processamento desnecessário de conteúdo irrelevante. Estas sub etapas são comuns aos processos de treinamento e de classificação e sumarização, tendo em vista que os mesmos tratamentos aplicados aos dados de treino devem ser aplicados aos dados à classificar para a obtenção de resultados consistentes. As operações realizadas são detalhadas nas seções seguintes.

4.3.2.1 Normalização dos dados

Os dados provenientes da etapa anterior, devido a sua origem, apresentam algumas características que dificultam o processamento. É comum à documentos encontrados no meio digital o emprego da linguagem em um formato que não corresponde a estrutura padrão da língua. No escopo de comentários de produtos, entre os erros mais comuns, podemos citar a utilização incorreta de letras maiúsculas, falta de pontuação, emprego de pontuação de forma excessiva, utilização de abreviações. Os tratamentos de normalização, representados pela

etapa 2 da figura 3 e etapa 12 da figura 4, realizados nos comentários são apresentados na sequência.

4.3.2.1.1 Conversão para letras minúsculas

Muitos são os usuários que fazem uso excessivo de letras em caixa alta em seus comentários. Utilizando este recurso na escrita de palavras e até mesmo frases completas. Este comportamento é problemático para o PLN, pois dificulta a definição correta da classe gramatical das palavras em questão. A transformação automática de todas as palavras do texto em caixa baixa facilita a etapa de *POS tagging* mas pode ser um problema para a identificação de entidades nomeadas. Contudo, esta abordagem foi a escolhida, devido a sua simplicidade e tendo em vista que a identificação de entidades não faz parte dos objetivos do trabalho proposto.

4.3.2.1.2 Remoção de URLs

Uma característica comum à comentários de revisão de produtos é a inclusão de URLs que direcionam os usuários à novos recursos disponíveis na internet. Estas URLs são irrelevantes considerando o objetivo de classificação e sumarização de comentários. Representando um problema devido aos caracteres de pontuação que apresentam em sua composição. Por este motivo, durante a implementação do protótipo, optou-se por removê-las dos dados analisados, através de uma expressão regular.

4.3.2.1.3 Conversão de quebras de linha em pontuação

Observou-se que muitos usuários substituem as pontuações finais por quebras de linha. Todavia, tais pontuações são importantes para a etapa de divisão dos comentários em sentenças, tendo em vista que a maioria dos algoritmos utilizam estes caracteres como delimitadores. Como a maior parte dos comentários trata de apenas um tópico e o modelo proposto considera a classificação a nível de sentença, o protótipo desenvolvido faz a substituição de todas as quebras de linha pelo caractere de ponto final a fim de melhorar a divisão realizada em etapa seguinte.

4.3.2.1.4 Correção de erros gramaticais comuns

O uso de abreviações, a falta de acentuação e o emprego de escrita de forma alternativa é comum no meio digital. No caso do português do Brasil há, por exemplo, a palavra “não”, que aparece muitas vezes grafada como “nao”, “ñ” e, até mesmo, “n”. Algumas outras palavras também possuem variações bem conhecidas. Esta característica também dificulta a etapa de *POS tagging*, fazendo com que tais palavras não tenham sua classe gramatical reconhecida pelo algoritmo de rotulagem.

A utilização de um algoritmo para substituição automática é possível através de técnicas como o cálculo de distância entre palavras, inferências estatísticas e uso de dicionários, mas esta não é uma tarefa trivial, tendo em vista que tais algoritmos geralmente tem como saída um conjunto de possíveis palavras corretas, contando com a ação de um usuário humano para concretizar as substituições. Optou-se então por realizar o mapeamento e a substituição, através de expressões regulares, de um conjunto conhecido de abreviações comumente utilizadas na internet e também de todas as palavras presentes nas listas de sentimentos e de *domain features*. Este mapeamento está detalhado no Quadro 4.

Quadro 4: Mapeamento de palavras incorretas e sua versão final

Forma incorreta	Forma corrigida	Forma incorreta	Forma corrigida	Forma incorreta	Forma corrigida
blz	beleza	acessivel	acessível	inutil	inútil
eh	é	assistencia	assistência	lamentavel	lamentável
msg	mensagem	audio	áudio	ma	má
mta	muita	bussola	bússola	memoria	memória
mt	muito	camera	câmera	miseravel	miserável
mto		cartao	cartão	otima	ótima
msm	mesmo	conexao	conexão	otimo	ótimo
n	não	desagradavel	desagradável	pessima	péssima
ñ		desprezivel	desprezível	pessimo	péssimo
nao		detestavel	detestável	pratico	prático
p/	para	dificil	difícil	rapida	rápida
pq	por que	duravel	durável	rapido	rápido
q	que	expansivel	expansível	resolucao	resolução

s	sim
tbm	também
tb	
td	tudo
vce	você
vc	

facil	fácil
horrivel	horrível
imprestavel	imprestável
insatisfatorio	insatisfatório
insuportavel	insuportável
inuteis	inúteis

resoluçao	
s.o.	sistema operacional
tecnica	técnica
terrivel	terrível
uteis	úteis
util	útil

4.3.2.1.5 Remoção de espaços e pontuações duplicadas

Outro problema comum em textos provenientes do meio digital é a repetição desnecessária de espaços e pontuações. Conforme discutido anteriormente, as pontuações são importantes nas etapas de PLN, e, por este motivo, foram utilizadas expressões regulares para substituir, de forma recursiva, espaços e pontuações duplicadas por uma única ocorrência dos mesmos. Desta forma, textos como “eu amei este celular!!!!!” são transformados em “eu amei este celular!”.

4.3.2.2 Divisão em sentenças

O modelo proposto prevê a análise dos comentários a nível de sentença, por isso se faz necessária uma etapa cujo objetivo é transformar um comentário no conjunto das sentenças que o compõem. Assim como as demais sub etapas do pré-processamento a etapa de divisão em sentenças é aplicada tanto no processo de treinamento, como etapa 3 da figura 3, quanto no processo de classificação e sumarização, representada pela etapa 13 da figura 4. Contudo, enquanto no primeiro a saída da etapa é um conjunto de sentenças associadas à nota do comentário no segundo a saída é apenas o conjunto de sentenças, sem nota associada, tendo em vista que a nota será atribuída no momento da classificação.

A forma mais simples de realizar a divisão do texto em sentenças é utilizando expressões regulares para definir os delimitadores de sentença. Contudo, esta abordagem pode apresentar diversas falhas. Tendo em vista que as expressões regulares serão independentes de contexto. No caso do português do Brasil, por exemplo, alguns caracteres de pontuação podem ser utilizados com outros objetivos dentro do texto. É o caso de abreviaturas em pronomes de tratamento, onde “senhor” pode ser escrito como “Sr.”.

Existe também a possibilidade de utilizar modelos computacionais treinados para realizar esta divisão no idioma alvo. Nesta abordagem conjuntos de dados são utilizados para realizar o treinamento do modelo, de modo que o mesmo esteja preparado para tratar tais peculiaridades provenientes do idioma.

Para o desenvolvimento do protótipo em questão utilizou-se a segunda abordagem através do modelo treinado para o português do Brasil, desenvolvido em python e disponibilizado pela biblioteca NLTK.

4.3.2.3 Filtro de sentenças inicial

O processamento de conteúdo irrelevante em um grande conjunto de dados pode ser indesejado. Tanto devido ao aumento no tempo de processamento quanto pelo risco de introduzir variáveis indesejadas no sistema. Por estes motivos o protótipo desenvolvido realiza uma avaliação das sentenças provenientes da etapa anterior, a fim de retirar do sistema aquelas cujo conteúdo não se mostra relevante.

Esta etapa de filtragem leva em consideração dois fatores. São eles o tamanho da sentença e a presença de ao menos uma *domain feature*. Entre estes fatores o de maior interesse é a presença de *domain feature*. A validação de tamanho mínimo foi introduzida para evitar a necessidade de análise da presença de *domain features* em sentenças com tamanho inferior a 3 caracteres, já que sentenças com tamanho muito pequeno já indicam a falta de presença das mesmas.

Para a análise da presença de *domain features* é verificado se a sentença contém ao menos um dos valores presentes na listagem obtida anteriormente, na etapa 4 e 14 das figuras 3 e 4, respectivamente. Desta forma sentenças do tipo “ok” e “comprei a duas semanas” são removidas do fluxo.

4.3.3 Processamento

A etapa de processamento inclui as tarefas de manipulação das sentenças, utilizando técnicas de PLN, a fim de permitir o treinamento e a posterior classificação de novas entradas. Assim como a etapa de pré-processamento, esta etapa possui sub etapas executadas tanto no processo de treinamento quanto no processo de classificação e sumarização. Entre estas sub etapas estão a tokenização das

sentenças e a rotulagem dos tokens com as suas respectivas classes gramaticais, a filtragem de sentenças sem qualificadores, e com um adicional no processo de treino onde também são levadas em consideração sentenças com notas contraditórias, e, por último, a geração dos *feature sets*, conjuntos de dados que serão utilizados no treinamento e na classificação. Além destas etapas o processo de treinamento também possui uma etapa responsável por criar as tuplas que serão utilizadas na etapa seguinte para treinar o classificador. Já o processo de classificação e sumarização possui etapas para classificar as sentenças e criar o sumário das *domain features*. As etapas citadas são explicadas nas seções seguintes.

4.3.3.1 Tokenização e *POS tagging*

A etapa de tokenização, seguida pela rotulagem dos tokens com suas classes gramaticais, chamada de *POS tagging*, que ocorre no passo 6 do processo de treinamento, representado pela figura 3 e no passo 16 do processo de classificação e sumarização, representado pela figura 4, são partes importantes de ambos os processos.

A tokenização consiste da divisão da sentença em tokens, ou seja, o conjunto de elementos que compõem a mesma, incluindo as palavras e os elementos de pontuação. Este é um processo que depende do idioma tratado, pois em diferentes idiomas as regras para separação dos itens na sentença pode variar. E por este motivo a tokenização no sistema desenvolvido foi realizada utilizando o tokenizador disponibilizado pela biblioteca NLTK configurado para o idioma português do Brasil.

Na etapa de *POS tagging* cada token gerado anteriormente será marcado com um rótulo que representa a sua classe gramatical dentro da sentença. Assim como a tokenização a etapa de *POS tagging* é dependente do idioma tratado, tendo em vista que as regras gramaticais possuem diferenças entre os idiomas.

Corporas sintaticamente anotados, do idioma desejado, podem ser utilizados para treinar um rotulador, ou *tagger*, no idioma em questão. O grande conjunto de dados provenientes de tais corporas garante ao *tagger* criado a capacidade de rotular novas entradas com boa precisão.

A biblioteca NLTK possui *taggers* criados para um conjunto de idiomas, mas não possui nenhum específico para o português. Contudo, a biblioteca disponibiliza

corporas em português do Brasil, o que torna possível o treinamento de um novo *tagger* para tal idioma. Esta abordagem foi escolhida como primeira opção no desenvolvimento do protótipo, contudo o resultado obtido não foi considerado satisfatório. Optou-se então pela busca de um *tagger* criado pela comunidade e entre as opções encontradas optou-se pelo Nltk-Tagger-Portuguese[24], que apresentou bons resultados e está disponibilizado de forma *open source* através da plataforma GitHub.

4.3.3.2 Filtro de sentenças

A etapa de filtro de sentenças ocorre após a tokenização e rotulagem das mesmas. Nesta etapa é verificado se no conjunto de tokens que compõem a sentença há a presença de ao menos um dos itens provenientes das listas de palavras de sentimentos.

Além desta verificação o processo de treinamento possui uma checagem adicional, que consiste em averiguar se a sentença possivelmente possui polaridade contraditória à do comentário. Tendo em vista que a nota do comentário é propagada para as sentenças, o sistema verifica se sentenças consideradas positivas possuem alguma palavra de sentimento negativa e se sentenças consideradas negativas possuem alguma palavra de sentimento positiva, o que indica uma possível contradição. Caso isso se confirme a sentença é descartada. Este comportamento foi adotado para evitar a inclusão de dados controversos no treinamento do classificador. O que poderia prejudicar o resultado final.

4.3.3.3 Geração de *feature set*

Infelizmente não é possível treinar e classificar as entradas utilizando as sentenças em sua forma original. Para isso se faz necessária a criação de *feature sets*, conjuntos de dados derivados das sentenças que serão utilizados para o treinamento do classificador e posterior classificação de novas sentenças. Isto ocorre na etapa 8 do processo de treinamento, representado pela figura 3, e na etapa 18 do processo de classificação e sumarização, representando pela figura 4. A

[24] https://github.com/fmaruki/Nltk-Tagger-Portuguese

etapa 8 apresenta um pequeno diferencial, pois os *feature sets* criados são acompanhados da classe correspondente a nota associada a sentença, onde notas maiores ou iguais a 3 são convertidas para o valor “positivo” e notas menores que 3 são convertidas para o valor “negativo”.

Assim como previsto no modelo, os *feature sets* são compostos por três fatores. São eles a presença de palavras com polaridade positiva, presença de palavras com polaridade negativa e presença de palavras de negação próximas as palavras de sentimentos. Estes atributos são gerados da seguinte forma:

- Presença de palavras com polaridade positiva (*has_positive_words*): A lista de palavras com polaridades positivas é utilizada como base para realizar uma busca na sentença. Caso alguma das palavras seja encontrada este atributo é marcado como verdadeiro;
- Presença de palavras com polaridade negativa (*has_negative_words*): A definição deste atributo se dá da mesma forma que o anterior, com a diferença que neste é utilizada a lista de palavras com polaridades negativa;
- Contém negação (*has_word_of_denial*): Caso a sentença possua uma palavra que indique polaridade o sistema verifica a existência, em até três palavras de distância, de uma palavra de negação, o que possivelmente indica uma inversão da polaridade, como na frase “Este celular não é bom”.

4.3.3.4 Treinamento do classificador

O treinamento do classificador é a última etapa do processo de treinamento, representado pela etapa 9 da figura 3, gerando como saída um modelo capaz de classificar novas entradas em uma das classes propostas (positivo e negativo). Essa etapa é realizada utilizando o conjunto de *feature sets* associados às suas classes, e os algoritmos de aprendizagem MaxEnt GIS, MaxEnt IIS, Naive Bayes e Decision Tree. Como já mencionado, optou-se por utilizar esses algoritmos devido a sua disponibilidade na biblioteca NLTK, o que possibilita a análise da qualidade dos algoritmos na resolução do problema identificado. Para cada um dos algoritmos citados é gerado um modelo de classificação que será utilizado posteriormente.

O conjunto de comentários, antes obtido e armazenado localmente (etapa 1 da Figura 3), foi dividido em 2 subconjuntos, um para treino e outro para teste dos modelos treinados. Foi criada a coluna “tags” na tabela do banco de dados onde foram armazenados os comentários. Essa coluna suporta uma lista de rótulos textuais, com o propósito de identificar os comentários de cada subconjunto e permitir reproduzir a mesma divisão em todas as sessões de teste. Para definir os subconjuntos, o conjunto original foi disposto em ordem aleatória. Desse conjunto desordenado, os comentários do grupo inicial, com 10% do total de comentários, foi rotulado como “test”, e o grupo restante (90%) foi rotulado como “training”.

Antes de treinar os classificadores, os dois subconjuntos são recuperados do banco de dados e submetidos ao pré-processamento, descrito na seção 4.3.2. Os subconjuntos pré-processados são derivados em pares de *feature set* e polaridade, sendo, o primeiro, um conjunto de atributos de cada sentença (seção 4.3.3.3); e o segundo, a classe do comentário, definida a partir da nota atribuída por seu autor. Esses pares são submetidos como entrada para a função de treino de cada classificador, conforme documentação do NLTK, as quais retornam os modelos gerados.

Enquanto as funções de treino dos algoritmos *Naive Bayes* e *Decision Tree* recebem apenas os pares como parâmetros, as funções do algoritmo MaxEnt aceitam outros parâmetros. No protótipo implementado foram especificados apenas os parâmetros “algorithm”, que indica o algoritmo específico; “trace”, que indica o nível de informações de diagnóstico produzidas durante o treino; e “max_iter”, que indica o número máximo de iterações que o algoritmo deve realizar. Para utilizar os dois algoritmos de *Maximum Entropy* implementados pelo biblioteca, a função de treino do classificador MaxEnt é executada duas vezes, tendo como parâmetro o algoritmo GIS, na primeira; o algoritmo IIS, na segunda; e, em ambas, os parâmetros “trace” e “max_iter”, com os valores 0 e 1000, respectivamente.

Os modelos retornados pelas funções de treino são então avaliados quanto a sua acurácia, que é a relação entre o número de exemplos corretamente classificados e o total de exemplos classificados, dado pela divisão do primeiro pelo segundo. A Tabela 1 apresenta a acurácia de cada modelo gerado.

Tabela 1 - Acurácia obtida em cada modelo treinado

Decision Tree	MaxEnt GIS	MaxEnt IIS	Naive Bayes
85,978836	85,802469	85,802469	85,802469

Os modelos probabilísticos, gerados pelos algoritmos Naive Bayes e MaxEnt, são avaliados também quanto à informatividade das features para o modelo, ou seja, a influência dos valores de cada atributo do *feature set* na inferência da classe de uma sentença. A Tabela 2 apresenta os pares *feature*-valor com maior informatividade.

Tabela 2 - Ganho de informação dos pares *feature*-valor em cada modelo probabilístico

			Algoritmo probabilístico		
Feature	***Valor***	***Classe***	***MaxEnt IIS***	***MaxEnt GIS***	***Naive Bayes***
has_negative_words	Sim	negativo	5,63	4,857	-
has_positive_words	Não	negativo	4,052	3,376	2,5
has_positive_words	Sim	positivo	2,843	2,465	-
has_negative_words	Não	positivo	0,885	0,789	1,2
has_word_of_denial	Não	positivo	0,674	0,605	1
has_word_of_denial	Sim	positivo	0,649	-	-
has_word_of_denial	Sim	negativo	-1,166	-0,905	2
has_word_of_denial	Não	negativo	-2,275	-1,941	-
has_positive_words	Não	positivo	-2,482	-2,089	-
has_negative_words	Não	negativo	-3,541	-2,972	-
has_negative_words	Sim	positivo	-	-	-
has_positive_words	Sim	negativo	-	-	-

Essas avaliações são realizadas por meio de funções do NLTK e os resultados são registrados no log do sistema, os quais foram utilizados para comparação dos classificadores. Esses modelos são utilizados na classificação das sentenças, que é descrita na seção 4.3.3.5.

4.3.3.5 Classificação e Sumarização

A etapa de classificação, representada pela etapa 19 da Figura 4, utiliza cada um dos modelos resultantes do processo de treinamento, em conjunto com os

feature sets derivados das sentenças na etapa 18, para classificar essas sentenças em uma das classes propostas (positivo ou negativo).

As sentenças classificadas são utilizadas na etapa de sumarização, representada pela etapa 20 da Figura 4. Nesta etapa são identificadas as *domain features* que compõem cada uma das sentenças, em seguida são identificados os comentários a que cada sentença pertence, e por último é realizada a contagem dos comentários de forma agrupada por *domain feature* e polaridade. O resultado destas operações é o total de comentários que avaliaram cada *domain feature* de forma positiva e negativa.

Os dados gerados são ordenados de forma decrescente pelo total de avaliações positivas e negativas associadas, tendo em vista que as *domain features* mais comentadas se mostram mais interessantes para um grande número de usuários. Os dados ordenados são exibidos como saída do processo em formato de sumário, contendo cada *domain feature* mencionada, seguida do total de avaliações positivas e negativas sobre ela.

4.4 AVALIAÇÃO

Nesta seção é descrito o método usado para avaliar o modelo proposto, através do protótipo implementado, quanto à sua eficácia na resolução do problema identificado e à sua precisão ao processar e classificar textos em linguagem natural. Em seguida, os resultados da avaliação são apresentados.

4.4.1 Método

Conforme apresentado na seção 1.2, um dos objetivos deste trabalho é desenvolver uma ferramenta capaz de analisar um conjunto de comentários, escritos em português do Brasil, e sumariza-los, levantando os principais pontos positivos e negativos comentados. Sendo assim, a avaliação do protótipo implementado direcionou-se para a análise da veracidade do sumário gerado pela ferramenta. Porém, como as tarefas de agrupamento e contagem das polaridades são triviais, e não são as mais significantes para a avaliação da eficácia e precisão do modelo, os dados foram analisados desagrupados, em uma estrutura pré-sumário (lista de

comentários associados às características contidas em cada um, e suas respectivas polaridades), permitindo cálculos mais precisos.

A capacidade do cérebro humano não foi superada pela computação, até então, quando se trata de interpretar as diversas sutilezas envolvidas na comunicação, tais como marcação de ênfase, delicadeza, dúvida, sarcasmos, informalidade e ausência de detalhes. Dessa forma, assumiu-se como resultado ideal a lista de comentários com as características identificadas e classificadas por um humano, aqui denominado sumário ideal, o qual foi usado como referência na análise do resultado do protótipo.

Para a elaboração do sumário ideal, foram selecionados aleatoriamente 100 comentários dentre os 10272 obtidos, e identificados através do rótulo “evaluation”, que foi inserido na lista de *tags* desses comentários no banco de dados local. Os 100 comentários de validação foram, então, analisados pelos 2 autores deste trabalho. Cada autor leu o conjunto de validação e, para cada comentário, identificou as características de celulares referenciadas, direta ou indiretamente, e a polaridade do sentimento expresso no texto sobre cada característica. Essas informações foram comparadas entre os autores e as diferenças discutidas para definir a melhor interpretação, formando uma terceira relação de features e polaridades para cada comentário, sendo esse o sumário ideal. Para fins de visualização neste documento, a relação obtida foi estruturada de forma agrupada (Tabela 3), apresentando apenas a quantidade de comentários que expressaram sentimento positivo e negativo sobre cada característica encontrada no conjunto.

Tabela 3 - Sumário ideal, usado como referência na avaliação do protótipo

	Quantidade de comentários	
Característica	**Positivos**	**Negativos**
dispositivo	48	2
custo	17	0
bateria	4	8
câmera	10	1
design	9	0
processador	4	5
usabilidade	6	0
assistência técnica	0	3

	Quantidade de comentários	
Característica	**Positivos**	**Negativos**
sistema operacional	2	0
tamanho	1	1
tela	2	0
wi-fi	1	1
áudio	1	1
galeria de fotos	0	1
hardware	1	0
resistência água	1	0

recursos	3	0
3g	0	2
armazenamento	2	0
gps	1	1
led notificação	0	2
marca	2	0
memória ram	2	0
peso	2	0

sensor batimentos	1	0
sensor biométrico	1	0
sensor proximidade	0	1
tamanho da tela	0	1
toques	0	1
touchscreen	0	1
vidro traseiro	0	1
viva-voz	0	1

O conjunto de 100 comentários para validação, foi submetido como entrada para o protótipo desenvolvido, que, após executar todas as etapas descritas na seção 4.3, apresentou como saída 4 sumários das features e polaridades encontradas, sendo que cada sumário foi gerado a partir do uso de um modelo diferente de classificação (MaxEnt GIS, MaxEnt IIS, Naive Bayes e Decision Tree). A Tabela 4 apresenta os sumários gerados pelo protótipo.

Tabela 4 - Sumário do sistema, por algoritmo de classificação

	Quantidade de comentários							
	Decision Tree		***MaxEnt GIS***		***MaxEnt IIS***		***Naive Bayes***	
Característica	**Pos.**	**Neg.**	**Pos.**	**Neg.**	**Pos.**	**Neg.**	**Pos.**	**Neg.**
dispositivo	48	1	48	1	48	1	48	1
câmera	13	0	13	0	13	0	13	0
sistema operacional	10	1	10	1	10	1	10	1
bateria	8	0	8	0	8	0	8	0
tela	5	1	5	1	5	1	5	1
processador	2	2	2	2	2	2	2	2
áudio	3	0	3	0	3	0	3	0
cor	3	0	3	0	3	0	3	0
marca	2	0	2	0	2	0	2	0
tamanho	2	0	2	0	2	0	2	0
3g	1	0	1	0	1	0	1	0
armazenamento	1	0	1	0	1	0	1	0
assistência técnica	1	0	1	0	1	0	1	0
atendimento	0	1	0	1	0	1	0	1
gps	1	0	1	0	1	0	1	0
resolução	1	0	1	0	1	0	1	0
tipo de tela	1	0	1	0	1	0	1	0

tv	1	0	1	0	1	0	1	0
wi-fi	1	0	1	0	1	0	1	0

A avaliação dos sumários gerados pelo protótipo foi realizada utilizando as métricas utilizadas em Salas-Zárate (2017), sendo elas:

- *Precision*, que representa a proporção de verdadeiros positivos (positivo no sumário ideal e no sumário do sistema) para o total de positivos do sistema.
- *Recall*, que representa a proporção de verdadeiros positivos para o total de positivos do sumário ideal.
- *F-score* (*F-measure*) é a média harmônica[25] dos valores de *precision* e *recall*.

Tais métricas são obtidas pelas seguintes fórmulas:

$$Precision = \frac{Verdadeiros\ Positivos}{Verdadeiros\ Positivos + Falsos\ Positivos}$$

$$Recall = \frac{Verdadeiros\ Positivos}{Verdadeiros\ Positivos + Falsos\ Negativos}$$

$$F - score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Assim como em Salas-Zárate (2017), as métricas foram utilizadas sobre dois aspectos do protótipo: (1) a capacidade de identificar características no texto e (2) a capacidade de identificar a polaridade do sentimento expresso no texto sobre cada característica.

Para avaliar a identificação de características, o F-score foi calculado comparando as características elencadas nos sumários gerados com as características do sumário ideal.

Para avaliar a classificação de sentimentos, o F-score foi obtido para cada característica, desconsiderando comentários onde a feature constou apenas no resultado ideal. Foi então calculada a média aritmética dos F-scores das features, obtendo-se o F-score total para cada um dos 4 sumários gerados pelo sistema.

[25] A média harmônica de dois números tende a ser mais próxima do menor deles, garantindo que, o valor do *F-score* só será alto, se o valor de *precision* e de *recall* forem altos.

4.4.2 Resultados

Aplicando-se o método apresentado na seção 4.4.1 ao protótipo desenvolvido, foram obtidos os resultados das Tabelas 5, 6 e 7. As Tabelas 5 e 6 apresentam os resultados quanto à precisão na identificação de características.

Tabela 5 - Relação entre as características do sumário ideal e as identificadas pelo sistema

	Sistema		
Sumário Ideal	*Identificadas*	*Não identificadas*	**Total**
Características	14	18	**32**
Não características	5	-	**5**
Total	**19**	**18**	

Para facilitar a interpretação da Tabela 6, a Tabela 5 apresenta a relação entre os valores reais (sumário ideal) e os valores preditos (sistema), com 14 verdadeiros positivos, 5 falsos positivos e 18 falsos negativos. Os verdadeiros negativos não foram contados, pois englobam todas as palavras que não são características, e esse valor é irrelevante para as métricas utilizadas.

Tabela 6 - Precision, recall e F-score das características identificadas pelo sistema

Precision	**Recall**	**F-score**
0,737	0,438	0,549

Os resultados na Tabela 5 indicam que a precisão do sistema na identificação de características foi de aproximadamente 74%. Isso significa que, de cada 100 características identificadas pelo sistema, 74 estão corretas, ou seja, realmente estão presentes e são alvos de sentimento no texto. Por outro lado, o *recall* não apresentou um bom resultado, pois indica que o sistema identificou menos da metade das características alvos de sentimento presentes no texto. Ou seja, mais da metade das características foram ignoradas juntamente com todas as expressões de sentimentos direcionadas a elas, o que pode reduzir significativamente a quantidade de informações úteis apresentadas no sumário gerado. O *F-score*, sendo a média harmônica, equilibrou os valores de *precision* e *recall*, ficando em aproximadamente 0,55.

A Tabela 7 apresenta os resultados quanto à precisão na classificação de polaridade.

Tabela 7 - Precision (P), recall (R) e F-score (F) das polaridades indicadas pelo sistema

	Naive Bayes			**Decision Tree**			**MaxEnt GIS**			**MaxEnt IIS**		
Característica	*P*	*R*	*F*	*P*	*R*	*F*	*P*	*R*	*F*	*P*	*R*	F
armazenamento	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
gps	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
processador	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
câmera	0,69	1,00	0,82	0,69	1,00	0,82	0,69	1,00	0,82	0,69	1,00	0,82
dispositivo	0,54	0,96	0,69	0,54	0,96	0,69	0,54	0,96	0,69	0,54	0,96	0,69
bateria	0,50	1,00	0,67	0,50	1,00	0,67	0,50	1,00	0,67	0,50	1,00	0,67
tamanho	0,50	1,00	0,67	0,50	1,00	0,67	0,50	1,00	0,67	0,50	1,00	0,67
áudio	0,33	1,00	0,50	0,33	1,00	0,50	0,33	1,00	0,50	0,33	1,00	0,50
sistema operacional	0,20	0,67	0,31	0,20	0,67	0,31	0,20	0,67	0,31	0,20	0,67	0,31
3g	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
assistência técnica	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
marca	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
tela	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
wi-fi	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Média	**0,41**	**0,62**	**0,48**	**0,41**	**0,62**	**0,48**	**0,41**	**0,62**	**0,48**	**0,41**	**0,62**	**0,48**

A Tabela 7 indica que não houve diferença significativa entre os algoritmos, com relação à classificação, apesar da vantagem de 0,17 na acurácia da classificação com Decision Tree. Além disso, ao contrário dos valores obtidos na Tabela 6, aqui o resultado de *recall* foi melhor que o de *precision*, indicando que os dados apresentados pelo sistema estão mais completos do que precisos. Ou seja, 62% das polaridades verdadeiramente positivas, foram classificadas corretamente pelo sistema (*recall*); mas, do total de ocorrências que o sistema classificou como positivas, apenas 41% eram realmente positivas (*precision*). O *F-score* da classificação de polaridades foi bastante próximo do *F-score* da identificação de características, se aproximando de 0,5.

5. CONCLUSÃO

Considerando o grande número de opiniões sobre produtos veiculadas na internet, e o esforço necessário para o consumidor ou fabricante analisarem tais informações, esse trabalho teve como objetivo a criação de um sistema computacional capaz de analisar comentários, em português, sobre um produto e resumir as opiniões expressas sobres as características do produto.

Um modelo para o sistema foi proposto, com base em técnicas e ferramentas geralmente utilizadas para o processamento de textos, e um protótipo foi implementado. O protótipo foi avaliado em um contexto delimitado e o resumo de opiniões gerado foi medido com base em métricas de comparação a um resumo considerado ideal.

Com os resultados obtidos, nota-se que a efetividade da identificação de características (Tabela 6) teve grande influência sobre os resultados da classificação de polaridade (Tabela 7), pois a classificação é feita apenas sobre as características identificadas, analisando as sentenças em que essas são referenciadas. Outro fator que influenciou a etapa de classificação foi a acurácia do classificador, que, apesar de não ter sido baixa para nenhum dos algoritmos, não foi suficiente para atingir uma boa precisão no resultado final do sistema. A acurácia poderia ser melhorada através da definição de um conjunto de características (*feature sets*) que melhor descrevam a polaridade da sentença, indo além da existência de palavras polarizadas, e abordando mais aspectos morfológicos e semânticos de textos que expressam polaridade.

Os resultados evidenciaram também que a abordagem de definir uma lista fixa de características a serem reconhecidas pelo sistema (*domain features*), limita a abrangência da etapa de extração de características do produto a partir do texto. Na grande quantidade de comentários podem ocorrer diversos sinônimos e palavras incomuns, descrevendo características do produto, ou até mesmo características novas, ainda não contidas na lista do sistema. Várias características desconhecidas, como essas, foram desconsideradas pelo protótipo implementado. Seguindo o mesmo raciocínio, a utilização de listas fixas de palavras polarizadas também

prejudicou os resultado obtidos, pois limitou a identificação de termos que expressam sentimentos.

Para validar os resultados obtidos, outros testes podem ser realizados sobre o modelo proposto e o protótipo implementado. Uma das possibilidades é comparar os sumário gerado pelo sistema com resultados gerados a partir de outras ferramentas de classificação de sentimentos a nível de aspectos, gerando uma espécie de benchmark para comparar a ferramenta aqui desenvolvida com outras disponíveis atualmente.

Por fim, conclui-se que o modelo proposto tem potencial para solução do problema identificado. Apesar de não ter atingido um bom nível de confiabilidade, ele apresenta um resumo qualitativo dos comentários processados, refletindo com certa precisão a opinião dos consumidores e servindo como auxílio em decisões de compra ou estratégias de mercado. O desenvolvimento deste trabalho, bem como as pesquisas envolvidas no processo, reafirmaram o grande potencial das áreas de mineração de textos, PLN e todas as outras vertentes de inteligência artificial. Os resultados obtidos foram, no geral, medianos, mas encorajadores para a evolução do modelo em trabalhos futuros.

5.1 TRABALHOS FUTUROS

Considerando o potencial do modelo proposto para a solução do problema identificado, os resultados deste trabalho e as limitações evidenciadas por esses, foram identificados algumas oportunidades de evolução do modelo. As possíveis melhorias são listadas a seguir, como sugestões para trabalhos futuros.

Alterar a forma como os comentários são divididos em unidades menores, de modo que a divisão não seja prejudicada pelo uso incorreto de pontuações (vírgulas ao invés de pontos, por exemplo). Uma possível solução seria criar unidades de sentimento, analisando morfo e sintaticamente as palavras próximas de cada característica do produto identificada no texto. Com tal análise, seriam descartadas as palavras que não expressam sentimentos e nem modificam as que expressam, formando pequenos grupos, apenas com as palavras que caracterizam o sentimento e seu alvo.

Utilizar técnicas mais abrangentes para a correção automática das palavras contidas nos comentários, ao invés do uso de expressões regulares pré-definidas. As expressões regulares se limitam às palavras e aos erros gramaticais previstos em sua construção, enquanto técnicas baseadas em dicionário abrangem um maior número de palavras e possíveis erros sobre elas.

Aumentar a cobertura da identificação de características do produto. Algumas possíveis soluções seriam: gerar a lista de características pré-definidas de forma automática, o que facilitaria a busca em diversas fontes; identificar as características com base no próprio conjunto de dados sendo processados, através de análise estatística ou semântica; ou consultar bases de informação estruturada (alternativas ao DBPedia Spotlight) capazes de informar os termos que descrevem uma entidade, que nesse contexto seria a entidade alvo dos comentários.

Utilizar uma fonte de dados com maior confiabilidade da relação entre o texto com a classificação do mesmo. Com o objetivo de evitar a incoerência apresentada em alguns comentários observados no estudo, onde o conteúdo se mostrava positivo, mas a nota aplicada pelo avaliador era negativa e vice versa. Uma possível abordagem para evitar este problema seria a classificação manual de um conjunto expressivo de comentários, criando uma base de dados para treinamento. Contudo, esta abordagem é problemática para tornar a ferramenta independente de contexto.

Expandir o conjunto de palavras que o modelo considera como expressões de sentimentos, utilizando léxicos de sentimentos. Ou ainda, ao invés de utilizar um conjunto pré-definido de palavras, identificar expressões de sentimento através de características textuais da sentença, a morfologia e a semântica de seus componentes, evitando que o uso de palavras incomuns prejudique essa tarefa.

Aumentar o ganho de informação com o *feature set* da classificação de polaridade, considerando características estruturais da sentença, além da presença de palavras que expressam sentimentos e palavras de negação. Avaliar o quanto características, como a posição das palavras de sentimento, dos adjetivos, das palavras de negação e a estrutura semântica da sentença, influenciam na polaridade da sentença e podem melhorar a precisão do classificador.

Utilizar outras técnicas de mineração de opiniões e PLN nas etapas do modelo, realizando novos testes e avaliando a evolução dos resultados. Exemplos

de outras técnicas são: *Pointwise Mutual Information (PMI)*, que é uma abordagem estatística para identificar palavras que traduzem opiniões; *Semantic-orientation Latent Semantic Analysis* (SO-LSA), que calcula a força da associação semântica entre dois termos, e pode ser utilizada na identificação de polaridade; *Latent Dirichlet Allocation* (LDA), muito utilizada para extração de tópicos em textos; e Lematização, que agrupa dentro de um lema as diferentes inflexões e variantes de um termo, e pode ser incluída na normalização dos dados, simplificando os termos que as etapas seguintes devem tratar.

REFERÊNCIAS

AGGARWAL, C. C.; ZHAI, C. **A survey of text clustering algorithms**. [S.l.]: [s.n.], 2012. V. 9781461432.

ARANHA, Christian; PASSOS, Emmanuel. A tecnologia de mineração de textos. **Revista Eletrônica de Sistemas de Informação**, Curitiba-PR, v. 5, n. 2, p.1-8, 2006. Disponível em: <http://www.spell.org.br/documentos/ver/26518/a-tecnologia-de-mineracao-de-textos/i/pt-br>. Acesso em: 22 ago. 2016.

ARCHAK, N.; GHOSE, A.; IPEIROTIS, P. G. Show me the Money ! Deriving the Pricing Power of Product. **Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '07**, p. 56–65, 2007.

BALAHUR, A.; MONTOYO, A. **Multilingual feature-driven opinion extraction and summarization from customer reviews**. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 2008.

BATISTA, Gustavo Enrique de Almeida Prado Alves. **Pré-processamento de dados em aprendizado de máquina supervisionado**. 2003. Tese (Doutorado em Ciências de Computação e Matemática Computacional) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2003. doi:10.11606/T.55.2003.tde-06102003-160219. Acesso em: 2017-03-20.

BECKER, Karin. **Introdução à Mineração de Opiniões**. In: Salgado, A.C.; Lóscio, B.F.. (Org.). Atualizações em Informática / XXXIV Congresso da Sociedade Brasileira de Computação. 1ed.Porto Alegre: SBC, 2014, v. 1, p. 125-175.

BHATTACHARYYA, P. **Natural Language Processing : A Perspective from Computation in Presence of Ambiguity , Resource Constraint and Multilinguality**. v. 1, n. 2, 2012.

CAMBRIA, E. et al. **New Avenues in Opinion Mining and Sentiment Analysis**. n. April, p. 15–21, 2013.

CHEN, C. et al. **Visual analysis of conflicting opinions**. IEEE Symposium on Visual Analytics Science and Technology 2006, VAST 2006 - Proceedings. 2006.

CHOWDHURY, G. 2003. **Natural language processing**. Annual Review of Information Science and Technology, ISSN 0066-4200. Disponível em: <http://dx.doi.org/10.1002/aris.1440370103>. Acesso em: 20 jul. 2016.

CONSUMIDOR. **Indicadores do Portal do Consumidor** - 1º Semestre 2015. Disponível em: <https://www.consumidor.gov.br/pages/dadosabertos/externo/>. Acessado em: 22 jun. 2016.

CONSUMIDOR. **Indicadores do Portal do Consumidor** - 1º Trimestre 2016. Disponível em: <https://www.consumidor.gov.br/pages/dadosabertos/externo/>. Acessado em: 22 jun. 2016.

CORDEIRO, L. G. T. **Geração automática de questões através de análise de texto**. [S.l.]: UFSC - Universidade Federal de Santa Catarina, 2016.

CORPUS in **Dicionário da Língua Portuguesa com Acordo Ortográfico**. Porto: Porto Editora, 2003-2016. Disponível em: <http://www.infopedia.pt/dicionarios/lingua-portuguesa/corpus>. Acesso em: 05 set. 2016.

DING, X.; LIU, B. **The utility of linguistic rules in opinion mining**. In:Proceedings ofthe 30th annual international ACM SIGIR conference on Research and developmentin information retrieval - SIGIR '07. [s.n.], 2007. p. 811. ISBN 9781595935977. ISSN 2151318X. Disponível em: <http://portal.acm.org/citation.cfm?doid=1277741-.1277921>.

ETZIONI, O. et al. **Web-scale information extraction in knowitall**. Proceedings of the 13th conference on World Wide Web - WWW '04. 2004. Disponível em: <http://portal.acm.org/citation.cfm?doid=988672.988687>

FERREIRA, Aurélio Buarque de Holanda. **Mini aurélio**: O dicionário da língua portuguesa. 7. ed. Curitiba: Positivo, 2008. 896 p. (ISBN 978-85-7472-959-6).

GANESAN, K.; ZHAI, C.; HAN, J. Opinosis : A Graph-Based Approach to Abstractive Summarization of Highly Redundant Opinions. **Proceedings of the 23rd International Conference on Computational Linguistics**, n. August, p. 340–348, 2010.

GENTLEMAN, R.; CAREY, V. J. Unsupervised machine learning. **Bioconductor case studies**. New York, NY: Springer New York, 2008, p. 137–157.

HAN, J.; KAMBER, M.; PEI, J. **Data mining: concepts and techniques**. 3ª ed. [S.l.]: Morgan Kaufmann, 2012.

HU, Minqing; LIU, Bing. **Mining and summarizing customer reviews**. Proceedings Of The 2004 Acm Sigkdd International Conference On Knowledge Discovery And Data Mining - Kdd '04, [S.l.], p.168-177, 2004a. ISBN 1581138889. ISSN 1581138889

HU, Minqing; LIU, Bing. Mining Opinion Features in Customer Reviews. **19th national conference on Artifical intelligence**, p. 755–760, 2004b.

HU, M.; LIU, B. Opinion extraction and summarization on the web. **Aaai**, p. 1–4, 2006.

JURAFSKY, Daniel; MARTIN, James H. **Speech and Language Processing**: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. v. 21, p. 0–934, 2009.

K, A. M.; ANTO, B. Ambiguities in natural language processing. **International journal of innovative research in computer and communication engineering (an iso certified organization)**, 2007. v. 32972, n. 5. Disponível em: <www.ijircce.com>.

KAROUSSI, Elham. Data Mining: K-Clustering Problem. Agder, Noruega. 2012.

KIM, H. C. *et al.* Constructing support vector machine ensemble. **Pattern recognition**, 2003. v. 36, n. 12, p. 2757–2767.

KIM, H. D.; ZHAI, C. **Generating comparative summaries of contradictory opinions in text**. Proceeding of the 18th ACM conference on Information and knowledge management - CIKM '09. 2009. Disponível em: <http://portal.acm.org/citation.cfm?doid=1645953.1646004>

KIM, H. D.; GANESAN, K. **Comprehensive review of opinion summarization**, 2011. p. 1–30. Disponível em: <http://www.ideals.illinois.edu/handle/2142/18702>.

KU, L. et al. Opinion Extraction, Summarization and Tracking in News and Blog Corpora. **Artificial Intelligence**, v. pages, n. 2001, p. 100–107, 2006.

LEWIS, M. Paul; SIMONS, Gary F.; FENNIG, Charles D. **Ethnologue:** Languages of the world. 19. ed. Dallas, TX: SIL international, 2009. Disponível em: <http://www.ethnologue.com>.

LIDDY, Elizabeth D. et al. Natural language processing. **Encyclopedia of library and information science**, v. 2, 2003. Disponível em: <http://www-nlpir.nist.gov/MINDS/FINAL/NLP.web.pdf>. Acesso em: 22 ago. 2016.

LINGUATECA. Linguateca. 2003. Disponível em: <http://www.linguateca.pt/>. Acesso em: 18 jun. 2017.

LIU, Bing; ZHANG, Lei. A survey of opinion mining and sentiment analysis. In: AGGARWAL, Charu C.; ZHAI, Chengxiang. **Mining text data**. New York: Springer US, 2012. Cap. 13. p. 415-463.

LIU, B. **Web Data Mining**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011.

LORENA, A. C.; CARVALHO, A. C. P. L. F. DE. Uma introdução às support vector machines. **Revista de informática teórica e aplicada**, 2007. v. 14, n. 2, p. 43–67. Disponível em: <http://seer.ufrgs.br/index.php/rita/article/viewArticle/rita_v14_n2_p43-67>.

LU, Y.; ZHAI, C. Opinion integration through semi-supervised topic modeling. **Proceeding of the 17th international conference on World Wide Web - WWW '08**, p. 121–130, 2008.

LU, Y.; ZHAI, C.; SUNDARESAN, N. Rated aspect summarization of short comments. **Proceedings of the 18th international conference on World wide web - WWW '09**, p. 131, 2009.

LUGER, George F. 2014. **Inteligência artificial**. 6. ed. São Paulo: Pearson Education do Brasil, c2014. xvii, 614 p. ISBN 9788581435503.

MEI, Q. et al. **Topic sentiment mixture**. Proceedings of the 16th international conference on World Wide Web - WWW '07. **Anais**...2007. Disponível em: <http://portal.acm.org/citation.cfm?doid=1242572.1242596>

MISHNE, G.; RIJKE, M. DE. MoodViews: Tools for blog mood analysis. **AAAI 2006 Spring Symposium on Computational Approaches to Analysing Weblogs**, p. 153–154, 2006.

MITCHELL, T. **Machine Learning**. New York, NY: McGraw-Hill, 1997. ISBN 0-07-042807-7.

NISHIKAWA, H. et al. Opinion Summarization with Integer Linear Programming Formulation for Sentence Extraction and Ordering. **Coling**, n. August, p. 910–918, 2010.

OLIVIER, C.; SCHÖLKOPF, B.; ZIEN, A. **Semi-supervised learning**. [S.l.]: [s.n.], 2006. V. 1.

POPESCU, A.-M.; ETZIONI, O. Extracting Product Features and Opinion from Reviews. **Human Language Technology and Empirical Methods in Natural Language Processing, Vancouver, British Columbia**, p. 339–346, 2005.

RICH, Elaine; KNIGHT, Kevin. **Inteligência artificial.** 2. ed. São Paulo: Makron Books, c1994. xxv, 722p. ISBN 8534601224 : (broch.)

SALAS-ZÁRATE, María Del Pilar et al. Sentiment Analysis on Tweets about Diabetes: An Aspect-Level Approach. **Computational and Mathematical Methods in Medicine**, [s. l.], v. 2017, 2017.

SARDINHA, Tony Berber. **Lingüística de Corpus**. São Paulo: Manoele, 2004. ISBN 85-204-1676-4.

SILVA, M., CARVALHO, P., SARMENTO, L. **Building a sentiment lexicon for social judgement mining**. Computational Processing of the Portuguese Language, 2012. p. 218–228.

SOUZA, M., VIEIRA, R., BUSETTI, D., CHISHMAN, R., ALVES, I. M. **Construction of a portuguese opinion lexicon from multiple resources**. In Proceedings of the 8th Brazilian Symposium in Information and Human Language Technology (STIL). SBC, 2011.

STOYANOV, V.; CARDIE, C. Partially Supervised Coreference Resolution for Opinion Summarization through Structured Rule Learning. **Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing**, n. July, p. 336–344, 2006a.

STOYANOV, V.; CARDIE, C. Toward opinion summarization: Linking the sources. **Proceedings of the Workshop on Sentiment and Subjectivity in Text - SST'06**, n. July, p. 9–14, 2006b.

STOYANOV, V.; CARDIE, C. Topic identification for fine-grained opinion analysis. **Proceedings of the 22nd International Conference on Computational Linguistics**, n. August, p. 817–824, 2008.

SUN, Ron. 2000. **Artificial Intelligence**: Connectionist and Symbolic Approaches. Disponível em: <http://www.cogsci.rpi.edu/~rsun/sun.encyc01.pdf>. Acessado: 15 jul. 2016.

TAN, P. N., STEINBACH, M., & KUMAR, V. (2006). **Introduction to Data Mining**. Pearson Addison Wesley.

TAUSCZIK, Y. R., PENNEBAKER, J.W. . **The psychological meaning of words**: Liwc and computerized text analysis methods. Journal of Language and Social Psychology, 29(1), 2010. p. 24–54.

TITOV, I.; MCDONALD, R. Modeling online reviews with multi-grain topic models. **Proceeding of the 17th international conference on World Wide Web (WWW)**, p. 111, 2008.

TSYTSARAU, M.; PALPANAS, T. **Survey on mining subjective data on the web**. Data Mining and Knowledge Discovery, v. 24, n. 3, p. 478–514, May 2012. ISSN 1573-756X.Disponível em:<http://dx.doi.org/10.1007/s10618-011-0238-6>.

VIEIRA, Renata; LOPES, Lucelene. Processamento de linguagem natural e o tratamento computacional de linguagens científicas. In: PERNA, Cristina Lopes;

DELGADO, Heloísa Koch; FINATTO, Maria José (Org.). **Linguagens especializadas em corpora:** Modos de dizer e interfaces de pesquisa. Porto Alegre: Edipucrs, 2010. p. 183-201. (ISBN 978-85-397-0024-0). Disponível em: <http://www.pucrs.br/edipucrs/linguagensespecializadasemcorpora.pdf>. Acesso em: 19 ago. 2016.

VISL. **Visual Interactive Syntax Learning**. 1996. Disponível em: <http://beta.visl.sdu.dk/>. Acesso em: 18 jun. 2017.

WIKIPEDIA. **NATURAL LANGUAGE PROCESSING**. 2016. Disponível em: <https://en.wikipedia.org/wiki/Natural_language_processing>. Acesso em: 20 jul. 2016.

ZHANG, J., ACKERMAN, M. S., ADAMIC, L. **Expertise networks in online communities**: structure and algorithms. In WWW ’07: Proceedings of the 16th international conference on World Wide Web. ACM, New York, NY, USA, 2007. p. 221–230.

ZHUANG, L., JING, F., ZHU, X.-Y. **Movie review mining and summarization**. In CIKM ’06: Proceedings of the 15th ACM international conference on Information and knowledge management. ACM, New York, NY, USA, 2006. p. 43–50.

APÊNDICE A - Testes com a ferramenta DBPedia Spotlight

O DBpedia Spotlight é, segundo trecho retirado da homepage da mesma, “uma ferramenta para anotação automática de menções de recursos da DBpedia em texto, provendo uma solução para vincular fontes de informação não estruturada à nuvem de Dados Abertos Conectados através da DBpedia”. Esta ferramenta é disponibilizada no formato de uma API REST, permitindo que textos sejam anotados com as informações encontradas como recursos na DBpedia. Desta forma, é possível analisar informações estruturadas de textos não estruturados. Além da API citada, a ferramenta disponibiliza uma aplicação de demonstração para que a comunidade possa realizar testes manuais dos serviços disponibilizados. Tal aplicação foi utilizada para verificar a possibilidade de detecção das características dos produtos a partir dos comentários. Um conjunto de dados, apresentado no Quadro 5, foi escolhido a partir do conjunto de dados de treino e utilizado em uma série de testes com diferentes configurações de parâmetros na ferramenta.

Quadro 5 - Comentários utilizados nos testes do DBpedia Spotlight

ID	Comentário
1	Com o positivo s480 tive a oportunidade de ter um aparelho bom e de custo baixo, e com ótimas funções, câmera de 8MP ótima para tirar fotos tanto em locais iluminados ou mais escuros, da para tirar ótimas selfies. Processador quad core roda a maioria dos jogos mas perde um pouco de qualidade gráfica, devido a tela ter resolução baixa, a bateria dura um tempo razoável assim como outros Smartphones.
2	Pesquisei muito antes de comprá-lo, entre diversos sites de tecnologia falando sobre smartphones com sistema android com bom custo benefício. Fiquei em dúvida entre 3 e escolhi esse, pela qualidade da câmera, suas diversas funções para fotos e também pelo sistema da asus, que facilita muito a navegação. Estou satisfeita com o produto. A bateria dura mais do que imaginava, tem como expandir espaço com cartão de memória (isso não foi necessário para mim, mas é bom para quem gosta de manter muitas músicas, vídeos e fotos no aparelho), o processamento é bom e tem como limpar a memória sem fechar os apps abertos. O som também é de qualidade. Único ponto negativo é a demora para carregar quando você continua utilizando o aparelho (cerca de 3h30 a 4h). Se deixá-lo parado, carrega em 2h.
3	O fato da Apple fazer seus próprios processadores, sistema operacional e placas, faz o sistema ser rápido e com excelência. A câmera é um diferencial, mesmo com desconfiados 8 mp, faz jus a fama de ótima captura de imagens, com destaque para os vídeos em movimento que não pedem o foco e também a câmera lenta. Felizmente ganhei meu Iphone 6 em um sorteio, logo então pensei em vendê-lo pelo seu alto valor, fato que nunca me fez querer comprar um, mas me surpreendi ao analisá-lo e resolvi ficar e usá-lo, não pelo status,

	mas por ser um diferencial em tecnologia.

O Quadro 6 apresenta os parâmetros utilizado nos testes bem como os resultados dos mesmos. Nele estão listadas apenas as palavras anotadas pela ferramenta. É possível perceber que a ferramenta não relacionou corretamente grande parte das palavras anotadas com recursos pertinentes ao contexto tratado. Aumentar o nível de confiança, além de não melhorar a relevância para o escopo, fez com que recursos anotados corretamente fossem removidos.

Quadro 6 - Conjunto de testes com a ferramenta DBpedia Spotlight

ID do comentário	Parâmetros		Resultado (palavra anotada)	Resource	Relevante para o escopo	Informação
	Confiança	Tipos				
1	0.5	Todos	baixo	http://pt.dbpedia.org/resource/Baixo	não	Instrumento musical
			Processador	http://pt.dbpedia.org/resource/Microprocessador	sim	Microprocessador
			bateria	http://pt.dbpedia.org/resource/Escola_de_samba	não	Escola de samba
	0.75	Todos	baixo	http://pt.dbpedia.org/resource/Baixo	não	Instrumento musical
2	0.5	Todos	asus	http://pt.dbpedia.org/resource/ASUS	não	Marca. Está correto, mas não se aplica ao escopo
			bateria	http://pt.dbpedia.org/resource/Tambor	não	Instrumento musical
			cartão de memória	http://pt.dbpedia.org/resource/Cart%C3%A3o_de_mem%C3%B3ria	sim	Cartão de memória

	0.75	Todos	asus	http://pt.dbpedia.org/resource/ASUS	não	Marca. Está correto, mas não se aplica ao escopo
3	0.5	Todos	Apple	http://pt.dbpedia.org/resource/Apple	não	Marca. Está correto, mas não se aplica ao escopo
			sistema operacional	http://pt.dbpedia.org/resource/Sistema_operativo	sim	Sistema operativo
			Iphone 6	http://pt.dbpedia.org/resource/IPhone_6	sim	Aponta para o recurso com informações do aparelho
	0.75	Todos	Apple	http://pt.dbpedia.org/resource/Apple	não	Marca. Está correto, mas não se aplica ao escopo
			Iphone 6	http://pt.dbpedia.org/resource/IPhone_6	sim	Aponta para o recurso com informações do aparelho

O Quadro 7 apresenta uma comparação entre as *domain features* identificadas utilizando o DBPedia Spotlight e a abordagem escolhida para a elaboração do protótipo desenvolvido.

Quadro 7 - Comparação de identificação de *domain features*

ID do comentário	Domain feature	Identificado corretamente pelo DBpedia spotlight	Identificado utilizando lista de domain features
1	câmera	Não	Sim
	aparelho	Não	Sim
	processador	Sim	Sim
	tela	Não	Sim
	bateria	Não	Sim
	smartphone	Não	Não
2	smartphone	Não	Não
	android	Não	Sim

	câmera	Não	Sim
	bateria	Não	Sim
	cartão de memória	Sim	Não
	memória	Não	Não
	som	Não	Sim
	processamento	Não	Sim
	aparelho	Não	Sim
3	processador	Não	Não
	sistema operacional	Sim	Não
	câmera	Não	Não

Através dos dados analisados é possível perceber que a identificação de *domain features* utilizando a DBpedia Spotlight não foi eficiente. Do total de 18 *domain features* presentes no conjunto de comentários testado somente três foram identificadas corretamente pela ferramenta.

APÊNDICE B - Código fonte desenvolvido

Arquivo README.md

```
# REVIEWS SUMMARIZATION #

### What is this repository for? ###

* Projeto para a manipulação dos comentários existentes, a fim de treinar uma rede de IA para classificar novas entradas

### How do I get set up? ###
* Instalar as dependências marcadas no requirements.txt

* Importar o nltk e executar nltk.download para baixar os corporas e pacotes da dependência

* O arquivo "database.sql" contem a estrutura do banco de dados
* O arquivo "dados.sql" contem os dados iniciais para popular a tabela de reviews
* O arquivo "Connection.py" aponta para o banco de dados
* O arquivo "example.py" é o ponto inicial de execução do projeto. Este arquivo também contém linhas de código utilizadas para testes
* O arquivo "reviewProcessor.py" possui as funções de tratamento de NLP. A função execute_workflow busca os comentários na tabela "en_review" e popula as tabelas "en_review_tokenizado" e "en_review_with_pos_tagging"
```

Arquivo requirements.txt

```
h5py==2.7.1
nltk==3.2.4
psycopg2==2.7.3.1
```

Arquivo .gitignore

```
# Created by .ignore support plugin (hsz.mobi)

.idea/

### Python template
# Byte-compiled / optimized / DLL files
__pycache__/
*.py[cod]
*$py.class

# Distribution / packaging
.Python
env/
build/
dist/

# Installer logs
pip-log.txt
pip-delete-this-directory.txt
```

Arquivo sql/database.sql

```
CREATE SCHEMA nlp;
CREATE TABLE IF NOT EXISTS nlp.en_review (
  id_review              SERIAL  NOT NULL PRIMARY KEY,
  url_pagina             TEXT,
  qtd_avaliacoes_inutil INTEGER DEFAULT 0,
  data_review            DATE,
  url_review             TEXT UNIQUE,
  qtd_avaliacoes_util    INTEGER DEFAULT 0,
  pontuacao_review       INTEGER NOT NULL,
  usuario_recomenda      BOOLEAN DEFAULT FALSE,
  conteudo_review        TEXT    NOT NULL,
  titulo_review          TEXT,
  CONSTRAINT en_review_unique UNIQUE (data_review, pontuacao_review,
    conteudo_review, titulo_review)
);

CREATE TABLE IF NOT EXISTS nlp.en_review_tokenizado (
  id_review_tokenizado       SERIAL          NOT NULL PRIMARY KEY,
  id_review                  INTEGER UNIQUE NOT NULL REFERENCES nlp.en_review
    (id_review),
  conteudo_review_tokenizado TEXT []         NOT NULL
);

CREATE TABLE IF NOT EXISTS nlp.en_review_with_pos_tagging (
  id_review_with_pos_tagging SERIAL          NOT NULL PRIMARY KEY,
  id_review                  INTEGER UNIQUE NOT NULL REFERENCES nlp.en_review
    (id_review),
  pos_tagging                TEXT []         NOT NULL
);

CREATE TABLE IF NOT EXISTS nlp.en_sentenca (
  id_sentenca SERIAL  NOT NULL PRIMARY KEY,
  ordem       INTEGER NOT NULL,
  id_review   INTEGER NOT NULL REFERENCES nlp.en_review (id_review)
);


CREATE TABLE IF NOT EXISTS nlp.en_sentenca_tokenizada (
  id_sentenca_tokenizada SERIAL          NOT NULL PRIMARY KEY,
  id_sentence            INTEGER UNIQUE NOT NULL REFERENCES nlp.en_review (
    id_review),
  sentenca_tokenizada    TEXT []         NOT NULL
);

CREATE TABLE IF NOT EXISTS nlp.en_sentenca_with_pos_tagging (
  id_sentenca_with_pos_tagging SERIAL          NOT NULL PRIMARY KEY,
  id_sentencea                 INTEGER UNIQUE NOT NULL REFERENCES nlp.
    en_review (id_review),
  pos_tagging                  TEXT []         NOT NULL
);

CREATE TABLE IF NOT EXISTS nlp.en_feature (
  id_feature SERIAL PRIMARY KEY,
  nome VARCHAR UNIQUE
);

ALTER TABLE nlp.en_review ADD COLUMN tags TEXT[] DEFAULT '{}';

ALTER TABLE nlp.en_feature ADD COLUMN feature_pai TEXT REFERENCES nlp.
    en_feature(nome);

--
-- DROP SCHEMA IF EXISTS nlp CASCADE
-- DROP TABLE IF EXISTS nlp.en_review_tokenizado;
-- DROP TABLE IF EXISTS nlp.en_review_with_pos_tagging;
-- DROP TABLE IF EXISTS nlp.en_feature;
```

Arquivo sql/dados_features.sql

```sql
INSERT INTO nlp.en_feature (nome, feature_pai)
VALUES
  ('carregador', NULL),
  ('peso', NULL),
  ('cor', NULL),
  ('banda', NULL),
  ('3g', NULL),
  ('4g', NULL),
  ('wi-fi', NULL),
  ('conexão', NULL),
  ('embalagem', NULL),
  ('assistência técnica', NULL),
  ('garantia', NULL),
  ('marca', NULL),
  ('memória ram', NULL),
  ('modelo', NULL),
  ('nfc', NULL),
  ('chip', NULL),
  ('recursos', NULL),
  ('resolução', NULL),
  ('atendimento', NULL),
  ('tv', NULL),

  -- AGRUPADAS
  -- armazenamento
  ('armazenamento', NULL),
  ('cartão de memória', 'armazenamento'),
  ('expansível', 'armazenamento'),
  ('memória interna', 'armazenamento'),

  -- processador
  ('processador', NULL),
  ('processamento', 'processador'),

  -- multichip
  ('multichip', NULL),
  ('dual chip', 'multichip'),
  ('dois chips', 'multichip'),

  -- camera
  ('câmera', NULL),
  ('câmera frontal', 'câmera'),
  ('câmera traseira', 'câmera'),

  -- tamanho
  ('tamanho', NULL),
  ('altura', 'tamanho'),
  ('largura', 'tamanho'),
  ('comprimento', 'tamanho'),

  -- tamanho da tela
  ('tamanho da tela', NULL),
  ('tamanho do display', 'tamanho da tela'),

  -- bateria
  ('bateria', NULL),
  ('tipo de bateria', 'bateria'),
  ('autonomia', 'bateria'),

  -- tipo de tela
  ('tipo de tela', NULL),
  ('lcd', 'tipo de tela'),
  ('oled', 'tipo de tela'),
  ('amoled', 'tipo de tela'),
  ('ips', 'tipo de tela'),

  -- sistema operacional
  ('sistema operacional', NULL),
  ('ios', 'sistema operacional'),
  ('android', 'sistema operacional');

```

```
-- features representando o dispositivo de forma geral
INSERT INTO nlp.en_feature (nome, feature_pai)
VALUES
  ('dispositivo', NULL),
  ('produto', 'dispositivo'),
  ('celular', 'dispositivo'),
  ('smartphone', 'dispositivo'),
  ('telefone', 'dispositivo'),
  ('aparelho', 'dispositivo');

-- features conhecidas (não retiradas pelo processo de crawler)
INSERT INTO nlp.en_feature (nome, feature_pai)
VALUES
  ('gps', NULL),
  ('áudio', NULL),
  ('microfone', NULL),
  ('bússola', NULL),
('tela', NULL),
('display', 'tela'),
('visor', 'tela'),
('som', 'áudio');
```

Arquivo example.py

```
from datetime import datetime

from review.review_processor import execute_workflow

time = datetime.now()

# Execucao do fluxo
execute_workflow()

print("Tempo de execucao: ", datetime.now() - time)
```

Arquivo dao/connection.py

```
import psycopg2
import psycopg2.extras

DB_URL = "dbname='postgres' " \
         "user='postgres' " \
         "host='localhost' " \
         "port='5432'" \
         "password='postgres'"


class Connection:
    conn = None

    def __init__(self):
        try:
            self.conn = psycopg2.connect(DB_URL)
        except Exception as e:
            print('Erro na conexao com o BD: {0}'.format(e))

    def execute_select(self, query, params):
        try:
            with self.conn:
                with self.conn.cursor() as cursor:
                    cursor.execute(query, params)
                    return cursor.fetchall()
        except Exception as e:
            print('Erro ao inserir no BD: {0}'.format(e))

    def close(self):
        self.conn.close()

    def execute_insert(self, query, params):
        try:
            with self.conn:
                with self.conn.cursor() as cursor:
                    cursor.execute(query, params)
        except Exception as e:
            print('Erro ao inserir no BD: {0}'.format(e))

    def execute_insert_returning_id(self, query, params):
        try:
            with self.conn:
                with self.conn.cursor() as cursor:
                    cursor.execute(query, params)
                    return cursor.fetchone()[0]
        except Exception as e:
            print('Erro ao inserir no BD: {0}'.format(e))
```

Arquivo review/constants.py

```
MIN_POSITIVE_RATING = 3
POSITIVE_POLARITY = 'positivo'
NEGATIVE_POLARITY = 'negativo'
ADJ = 'ADJ'
ADV = 'ADV'
RELEVANT_TAGS = [ADJ, ADV]
```

Arquivo dto/domain_feature.py

```
class DomainFeature:
    nome = None
    feature_pai = None

    def __init__(self, nome, feature_pai):
        self.nome = nome
        self.feature_pai = feature_pai
```

Arquivo review/feature_extractor.py

```
from review.constants import ADJ, ADV


POSITIVE_SENTIMENT_WORDS = ['acessível', 'adorei', 'amei', 'boa', 'boas', 'bom', 'bonito', 'contente', 'durável',
                            'eficiente', 'excelente', 'extraordinário', 'feliz', 'forte', 'fácil', 'gostei',
                            'legal', 'magnífico', 'maravilhosa', 'maravilhoso', 'melhor', 'perfeito', 'positivo',
                            'prático', 'rápida', 'rápido', 'satisfeito', 'superior', 'surpreendente', 'ótima',
                            'ótimo', 'útil']

NEGATIVE_SENTIMENT_WORDS = ['desagradável', 'desprezível', 'detestável', 'errado', 'fajuto', 'feio', 'fraca', 'fraco',
                            'horrível', 'imprestável', 'incorreto', 'inferior', 'insatisfatório', 'insatisfeito',
                            'insuportável', 'inúteis', 'inútil', 'lamentável', 'lento', 'maldito', 'miserável',
                            'má', 'negativo', 'ordinário', 'pior', 'podre', 'péssima', 'péssimo', 'ruim', 'terrível']


def get_positive_sentiment_words():
    return POSITIVE_SENTIMENT_WORDS


def get_negative_sentiment_words():
    return NEGATIVE_SENTIMENT_WORDS


def has_positive_word(tagged_text):
    for tagged_word in tagged_text:
        if tagged_word[0] in get_positive_sentiment_words():
            return True
    return False


def has_negative_word(tagged_text):
    negative_words = get_negative_sentiment_words()
    for tagged_word in tagged_text:
        if tagged_word[0] in negative_words:
            return True
    return False


def has_sentiment_word(tagged_text):
    for tagged_word in tagged_text:
        if tagged_word[1] == ADJ:
            return True
    return False


def has_word_of_denial(tagged_text):
    if ('não', ADV) in tagged_text:
        denial_index = tagged_text.index(('não', ADV))
        if denial_index > -1:
            list_after_denial = tagged_text[denial_index:]
            item_index = 0
            while item_index < len(list_after_denial):
                if list_after_denial[item_index][1] == ADJ:
                    return item_index < 3
                item_index += 1

    return False


def get_sentiment_words(tagged_text, sentiment_set):
    sentiment_words = []
    for (tagged_word, tag) in tagged_text:
        # tagged_word = tagged_tuple[0]
```

```
64          if tagged_word in sentiment_set:
65              sentiment_words.append(tagged_word)
66      sentiment_set.sort()
67      return sentiment_set
68
69
70  def get_positive_words(tagged_text):
71      return get_sentiment_words(tagged_text, get_positive_sentiment_words())
72
73
74  def get_negative_words(tagged_text):
75      return get_sentiment_words(tagged_text, get_negative_sentiment_words())
76
77
78  def get_features(tagged_text):
79      features = {
80          "has_positive_words": has_positive_word(tagged_text),
81          "has_negative_words": has_negative_word(tagged_text),
82          "has_word_of_denial": has_word_of_denial(tagged_text)
83      }
84      return features
```

Arquivo dao/features_dao.py

```
from typing import List

from dao.connection import Connection
from dto.domain_feature import DomainFeature


def get_domain_features() -> List[DomainFeature]:
    conn = Connection()
    query = 'SELECT nome, feature_pai FROM nlp.en_feature;'

    features = conn.execute_select(query, None)
    conn.close()

    if not features:
        raise ValueError("Não há features do domínio cadastradas. "
                         "Crie a tabela de features (database.sql) e execute o SQL do arquivo dados_features.sql")
    return [DomainFeature(feature[0], feature[1]) for feature in features]
```

Arquivo dao/review_dao.py

```
from dao.connection import Connection
from review.dto.review_object import Review


def get_all_reviews(tags=None):
    if tags is None:
        tags = ["test", "training"]
    conn = Connection()
    select_all_reviews_query = """
        SELECT id_review, url_pagina, qtd_avaliacoes_inutil, data_review, url_review, qtd_avaliacoes_util,
          pontuacao_review, usuario_recomenda, conteudo_review, titulo_review
        FROM nlp.en_review
        WHERE tags @> ARRAY[%s]
        ORDER BY id_review;
"""

    reviews_result = conn.execute_select(select_all_reviews_query, tags)
    reviews = list()
    if reviews_result is not None:
        for review_result in reviews_result:
            reviews.append(
                Review(
                    review_result[0], review_result[1], review_result[2], review_result[3], review_result[4],
                    review_result[5], review_result[6], review_result[7], review_result[8], review_result[9]
                )
            )
    return reviews
```

Arquivo example.py

```
class Review:
    id_review = None
    url_pagina = None
    qtd_avaliacoes_inutil = None
    data_review = None
    url_review = None
    qtd_avaliacoes_util = None
    pontuacao_review = None
    usuario_recomenda = None
    conteudo_review = None
    titulo_review = None

    def __init__(self, _id_review, _url_pagina, _qtd_avaliacoes_inutil, _data_review, _url_review, _qtd_avaliacoes_util,
                 _pontuacao_review, _usuario_recomenda, _conteudo_review, _titulo_review):
        self.id_review = _id_review
        self.url_pagina = _url_pagina
        self.qtd_avaliacoes_inutil = _qtd_avaliacoes_inutil
        self.data_review = _data_review
        self.url_review = _url_review
        self.qtd_avaliacoes_util = _qtd_avaliacoes_util
        self.pontuacao_review = _pontuacao_review
        self.usuario_recomenda = _usuario_recomenda
        self.conteudo_review = _conteudo_review
        self.titulo_review = _titulo_review
```

Arquivo review_processor.py

```
import nltk

from dao.features_dao import get_domain_features
from dao.review_dao import get_all_reviews
from nlp.tagger import get_pos_tagger, get_sent_tokenizer
from review.constants import MIN_POSITIVE_RATING, NEGATIVE_POLARITY,
    POSITIVE_POLARITY
from review.feature_extractor import get_features
from review.review_util import get_sanitized_reviews,
    has_adverbs_or_adjectives, has_contradictory_word, \
    is_content_useful
from review.summarizer import summarize_and_print

IIS = "IIS"
GIS = "GIS"


def tokenize(content):
    return nltk.word_tokenize(content, 'portuguese')


def pos_tagging(tagger, content):
    return tagger.tag(content)


def get_feature_tuple(sentence_with_pos_tagging, review):
    features_of_sentence = get_features(sentence_with_pos_tagging)
    rating = NEGATIVE_POLARITY
    if review.pontuacao_review > MIN_POSITIVE_RATING:
        rating = POSITIVE_POLARITY

    return features_of_sentence, rating


def execute_workflow():
    reviews_train = get_all_reviews(["training"])
    reviews_test = get_all_reviews(["test"])
    domain_features = get_domain_features()

    train_set = preprocess(reviews_train, domain_features)
    test_set = preprocess(reviews_test, domain_features)

    models = create_models(test_set, train_set)
    classify_and_summarize(models, test_set, domain_features)


def preprocess(reviews, domain_features):
    tagger = get_pos_tagger()
    sent_tokenizer = get_sent_tokenizer()
    preprocessed_sentences = []

    for review in get_sanitized_reviews(reviews, domain_features):
        for order, sentence in enumerate(sent_tokenizer.tokenize(review.
    conteudo_review)):

            if is_content_useful(sentence, domain_features):
                tokenized_sentence = tokenize(sentence)
                sentence_with_pos_tagging = pos_tagging(tagger,
    tokenized_sentence)

                if has_adverbs_or_adjectives(sentence_with_pos_tagging) \
                        and not has_contradictory_word(
    sentence_with_pos_tagging, review.pontuacao_review):
                    preprocessed_sentences.append((sentence_with_pos_tagging,
    review))

    return preprocessed_sentences


def create_models(test_set, train_set):
    models = list()
```

```
66      train_feature_tuples = [get_feature_tuple(sentence_pos, review) for
        sentence_pos, review in train_set]
67      test_feature_tuples = [get_feature_tuple(sentence_pos, review) for
        sentence_pos, review in test_set]
68
69      naive_bayes_classifier = nltk.NaiveBayesClassifier.train(
        train_feature_tuples)
70      models.append((naive_bayes_classifier, "Naive Bayes"))
71
72      decision_tree_classifier = nltk.DecisionTreeClassifier.train(
        train_feature_tuples)
73      models.append((decision_tree_classifier, "DecisionTree"))
74
75      maxent_gis_classifier = nltk.classify.MaxentClassifier.train(
        train_feature_tuples, GIS, trace=0, max_iter=1000)
76      models.append((maxent_gis_classifier, "MaxEnt GIS"))
77
78      maxent_iis_classifier = nltk.classify.MaxentClassifier.train(
        train_feature_tuples, IIS, trace=0, max_iter=1000)
79      models.append((maxent_iis_classifier, "MaxEnt IIS"))
80
81      for classifier, name in models:
82          test_classifier(classifier, name, test_feature_tuples)
83
84      return models
85
86
87  def test_classifier(classifier, name, test_feature_tuples):
88      print("{} acurracy -> {}".format(name, nltk.classify.accuracy(classifier,
        test_feature_tuples)))
89
90      if hasattr(classifier, "show_most_informative_features"):
91          print("{} most informative features -> ".format(name))
92          print(classifier.show_most_informative_features())
93
94      return classifier
95
96
97  def classify_and_summarize(models, test_set, domain_features):
98      for classifier, name in models:
99          print("\n{} Summarization . . .".format(name))
100         summarize_and_print(test_set, classifier, domain_features)
```

Arquivo review/dto/review_summary.py

```
class ReviewSummary:
    domain_feature = None
    positive = 0
    negative = 0
    total = 0

    def __init__(self, _domain_feature, _positive=0, _negative=0):
        self.domain_feature = _domain_feature
        self.positive = _positive
        self.negative = _negative
        self.total = _positive + _negative

    def __str__(self) -> str:
        # imprime sinais positivo e negativo em unicode
        return "{}:\n\t\u2795 {} \t\u2796 {} \tTotal -> {}\n" \
            .format(self.domain_feature, self.positive, self.negative, self.
    total)
```

Arquivo review/review_util.py

```
import re
from typing import List

from dto.domain_feature import DomainFeature
from review.constants import MIN_POSITIVE_RATING, RELEVANT_TAGS
from review.feature_extractor import has_positive_word, has_negative_word


def remove_unnecessary_spaces(sentence):
    return re.sub(r"\s([.!,?])", r"\1", sentence)


def remove_double_punctuation(sentence):
    return re.sub(r"([.!,?\'\"\s])\1+", r"\1", sentence)


def replace_misspelled_common_words(sentence):
    sentence = re.sub(r'\sblz\s', ' beleza ', sentence)
    sentence = re.sub(r'\s?eh([\s,])', r' é\1', sentence)
    sentence = re.sub(r'\smsg\s', ' mensagem ', sentence)
    sentence = re.sub(r'\smta\s', ' muita ', sentence)
    sentence = re.sub(r'\smto?\s', ' muito ', sentence)
    sentence = re.sub(r'\smsm\s', ' mesmo ', sentence)
    sentence = re.sub(r'\s[n ]\s', ' não ', sentence)
    sentence = re.sub(r'\s?nao\s', ' não ', sentence)
    sentence = re.sub(r'\sp/\s', ' para ', sentence)
    sentence = re.sub(r'\spq\s', ' por que ', sentence)
    sentence = re.sub(r'\sq\s', ' que ', sentence)
    sentence = re.sub(r'\ss\s', ' sim ', sentence)
    sentence = re.sub(r'\stbm?\s', ' também ', sentence)
    sentence = re.sub(r'\std\s', ' tudo ', sentence)
    sentence = re.sub(r'\svce?\s', ' você ', sentence)

    sentence = re.sub(r'\sacessivel\s', ' acessível ', sentence)
    sentence = re.sub(r'\sassistencia\s', ' assistência ', sentence)
    sentence = re.sub(r'\saudio\s', ' áudio ', sentence)
    sentence = re.sub(r'\sbussola\s', ' bússola ', sentence)
    sentence = re.sub(r'\scamera\s', ' câmera ', sentence)
    sentence = re.sub(r'\scartao\s', ' cartão ', sentence)
    sentence = re.sub(r'\sconexao\s', ' conexão ', sentence)
    sentence = re.sub(r'\sdesagradavel\s', ' desagradável ', sentence)
    sentence = re.sub(r'\sdesprezivel\s', ' desprezível ', sentence)
    sentence = re.sub(r'\sdetestavel\s', ' detestável ', sentence)
    sentence = re.sub(r'\sdificil\s', ' difícil ', sentence)
    sentence = re.sub(r'\sduravel\s', ' durável ', sentence)
    sentence = re.sub(r'\sexpansivel\s', ' expansível ', sentence)
    sentence = re.sub(r'\sfacil\s', ' fácil ', sentence)
    sentence = re.sub(r'\shorrivel\s', ' horrível ', sentence)
    sentence = re.sub(r'\simprestavel\s', ' imprestável ', sentence)
    sentence = re.sub(r'\sinsatisfatorio\s', ' insatisfatório ', sentence)
    sentence = re.sub(r'\sinsuportavel\s', ' insuportável ', sentence)
    sentence = re.sub(r'\sinuteis\s', ' inúteis ', sentence)
    sentence = re.sub(r'\sinutil\s', ' inútil ', sentence)
    sentence = re.sub(r'\slamentavel\s', ' lamentável ', sentence)
    sentence = re.sub(r'\sma\s', ' má ', sentence)
    sentence = re.sub(r'\smemoria\s', ' memória ', sentence)
    sentence = re.sub(r'\smiseravel\s', ' miserável ', sentence)
    sentence = re.sub(r'\sotim([ao])\s', r' ótim\1 ', sentence)
    sentence = re.sub(r'\spessim([ao])\s', r' péssim\1 ', sentence)
    sentence = re.sub(r'\spratico\s', ' prático ', sentence)
    sentence = re.sub(r'\srapid([ao])\s', r' rápid\1 ', sentence)
    sentence = re.sub(r'\sresolu[cç]ao\s', ' resolução ', sentence)
    sentence = re.sub(r'\ss\.o\.\s', ' sistema operacional ', sentence)
    sentence = re.sub(r'\stecnica\s', ' técnica ', sentence)
    sentence = re.sub(r'\sterrivel\s', ' terrível ', sentence)
    sentence = re.sub(r'\suteis\s', ' úteis ', sentence)
    sentence = re.sub(r'\sutil\s', ' útil ', sentence)

    sentence = re.sub(r'\.', '. ', sentence)
    sentence = re.sub(r',', ', ', sentence)
    return sentence
```

```
72
73
74 def convert_break_lines_into_punctuation(sentence):
75     original_sentence = sentence
76     sentence = sentence.replace('\n\n', '\n').replace('\r', '')
77     if original_sentence == sentence:
78         return sentence.replace('\n', '. ').replace('\r', '')
79     else:
80         return convert_break_lines_into_punctuation(sentence)
81
82
83 def remove_links_from_content(content):
84     return re.sub(r'https?:\/\/\S*', '', content, flags=re.MULTILINE)
85
86
87 def has_relevant_length(content):
88     return not content.isspace() and len(content) > 2
89
90
91 def has_domain_features(content, domain_features: List[DomainFeature]):
92     for feature in domain_features:
93         if feature.nome in content:
94             return True
95     return False
96
97
98 def is_content_useful(content, domain_features: List[DomainFeature]):
99     return has_relevant_length(content) and has_domain_features(content,
    domain_features)
100
101
102 def has_contradictory_word(sentence, score):
103     is_contradictory = has_negative_word(sentence) if score >
    MIN_POSITIVE_RATING else has_positive_word(sentence)
104
105     if is_contradictory:
106         print("sentença contraditória score -> {} | sentence -> {}".format(
    score, sentence))
107
108     return is_contradictory
109
110
111 def has_adverbs_or_adjectives(sentence_with_pos_tagging):
112     for (_, tag) in sentence_with_pos_tagging:
113         if tag in RELEVANT_TAGS:
114             return True
115     return False
116
117
118 def print_removed_reviews(reviews_tuple):
119     print('------------ Reviews removidos (INICIO) ------------')
120     for (review, cause) in reviews_tuple:
121         print('id_review -> {} || causa -> {} || conteudo -> {}'
122               .format(review.id_review, cause, review.conteudo_review))
123     print('------------ Reviews removidos (FIM) ------------')
124
125
126 def get_sanitized_reviews(reviews, domain_features: List[DomainFeature]=list()
    , filter_by_domain_features=True):
127     sanitized_reviews = []
128     removed_reviews = []
129     if len(reviews) != 0:
130         for review in reviews:
131             lower_content = review.conteudo_review.lower()
132             # Links afetam a tokenização
133             sanitized_content = remove_links_from_content(lower_content)
134             # Quebras de linha afetam a quebra em sentenças
135             sanitized_content = convert_break_lines_into_punctuation(
    sanitized_content)
136             # Palavras digitadas inoorretamente comprometem a geração das tags
137             sanitized_content = replace_misspelled_common_words(
    sanitized_content)
```

```
138             # Remove espaços antes de pontuação
139             sanitized_content = remove_unnecessary_spaces(sanitized_content)
140             # Pontuações duplas afetam a tokenização e a quebra em senteças
141             sanitized_content = remove_double_punctuation(sanitized_content)
142 
143             # Só devem ser mantidos reviews com conteúdo útil para ser
    analisado
144             if not filter_by_domain_features or is_content_useful(
    sanitized_content, domain_features):
145                 review.conteudo_review = sanitized_content
146                 sanitized_reviews.append(review)
147             else:
148                 removed_reviews.append((review, 'Conteúdo desprezível'))
149 
150     print_removed_reviews(removed_reviews)
151     print("Total de reviews -> {}".format(len(sanitized_reviews)))
152     return sanitized_reviews
```

Arquivo review/summarizer.py

```
from typing import List, Set

from dto.domain_feature import DomainFeature
from review.constants import POSITIVE_POLARITY, NEGATIVE_POLARITY
from review.dto.review_summary import ReviewSummary
from review.feature_extractor import get_features


def summarize_and_print(sentence_review_tuples, classifier, domain_features: List[DomainFeature]):
    reviews_by_domain_feature = group_by_feature_and_polarity(sentence_review_tuples, domain_features, classifier)
    print_summarization(summarize(reviews_by_domain_feature))


def group_by_feature_and_polarity(sentence_review_tuples, domain_features: List[DomainFeature], classifier):
    reviews_by_domain_feature = {}
    for sentence_with_pos_tag, review in sentence_review_tuples:
        polarity = classifier.classify(get_features(sentence_with_pos_tag))
        domain_features_in_sentence = extract_domain_features(sentence_with_pos_tag, domain_features)

        for domain_feature in domain_features_in_sentence:
            feature_key = domain_feature.feature_pai if domain_feature.feature_pai else domain_feature.nome
            if feature_key not in reviews_by_domain_feature:
                reviews_by_domain_feature[feature_key] = {POSITIVE_POLARITY: [], NEGATIVE_POLARITY: []}
            reviews_by_domain_feature[feature_key][polarity].append(review.id_review)
    return reviews_by_domain_feature


def extract_domain_features(sentence_with_pos_tag, domain_features: List[DomainFeature]) -> Set[DomainFeature]:
    features_in_sentence = get_domain_features(sentence_with_pos_tag, domain_features)
    return set(filter((lambda feature: feature is not None), features_in_sentence))


def get_domain_features(sentence_with_pos_tag, domain_features: List[DomainFeature]):
    words = [word for word, tag in sentence_with_pos_tag]
    sentence = " ".join(words)
    features_in_sentence = [feature for feature in domain_features if feature.nome in sentence]

    return features_in_sentence


def summarize(reviews_by_domain_feature) -> List[ReviewSummary]:
    summarization = []
    # conta reviews, nao sentencas
    for domain_feature, reviews_by_polarity in reviews_by_domain_feature.items():
        positive_reviews = set(reviews_by_polarity[POSITIVE_POLARITY])
        negative_reviews = set(reviews_by_polarity[NEGATIVE_POLARITY])
        summarization.append(ReviewSummary(domain_feature, len(positive_reviews), len(negative_reviews)))
    return summarization


def print_summarization(summarization: List[ReviewSummary]):
    n_features, n_reviews, n_positives, n_negatives = 0, 0, 0, 0

    print("\n--- SUMARIZACAO ---")
    for review_summary in sorted(summarization, key=lambda rs: rs.total, reverse=True):
        n_positives += review_summary.positive
```

```
57          n_negatives += review_summary.negative
58          n_features += 1
59          n_reviews += review_summary.total
60          print(review_summary.__str__())
61
62      print("TOTAL:\n\tPOSITIVOS: {}\n\tNEGATIVOS: {}"
63            "\n\tFEATURES: {}\n\tREVIEWS: {}".format(n_positives, n_negatives,
    n_features, n_reviews))
```

Arquivo /dto/tagged_word.py

```
class TaggedWord:
    word = None
    lemma = None
    tag = None

    def __init__(self, word, lemma, tag):
        self.word = word
        self.lemma = lemma
        self.tag = tag
```

Arquivo thirdpartyprojects/tagger/fmaruky/tagger.py

```
# https://github.com/fmaruki/Nltk-Tagger-Portuguese
import pickle
from random import shuffle
from string import punctuation

import nltk


def convert_to_universal_tag(t, reverse=False):
    tagdict = {
        'n': "NOUN",
        'num': "NUM",
        'v-fin': "VERB",
        'v-inf': "VERB",
        'v-ger': "VERB",
        'v-pcp': "VERB",
        'pron-det': "PRON",
        'pron-indp': "PRON",
        'pron-pers': "PRON",
        'art': "DET",
        'adv': "ADV",
        'conj-s': "CONJ",
        'conj-c': "CONJ",
        'conj-p': "CONJ",
        'adj': "ADJ",
        'ec': "PRT",
        'pp': "ADP",
        'prp': "ADP",
        'prop': "NOUN",
        'pro-ks-rel': "PRON",
        'proadj': "PRON",
        'prep': "ADP",
        'nprop': "NOUN",
        'vaux': "VERB",
        'propess': "PRON",
        'v': "VERB",
        'vp': "VERB",
        'in': "X",
        'prp-': "ADP",
        'adv-ks': "ADV",
        'dad': "NUM",
        'prosub': "PRON",
        'tel': "NUM",
        'ap': "NUM",
        'est': "NOUN",
        'cur': "X",
        'pcp': "VERB",
        'pro-ks': "PRON",
        'hor': "NUM",
        'pden': "ADV",
        'dat': "NUM",
        'kc': "ADP",
        'ks': "ADP",
        'adv-ks-rel': "ADV",
        'npro': "NOUN",
    }
    if t in ["N|AP", "N|DAD", "N|DAT", "N|HOR", "N|TEL"]:
        t = "NUM"
    if reverse:
        if "|" in t: t = t.split("|")[0]
    else:
        if "+" in t: t = t.split("+")[1]
        if "|" in t: t = t.split("|")[1]
        if "#" in t: t = t.split("#")[0]
    t = t.lower()
    return tagdict.get(t, "." if all(tt in punctuation for tt in t) else t)


# nltk.corpus.mac_morpho.tagged_sents is incorrect, converting tagged_paras to tagged_sents
dataset1 = list(nltk.corpus.floresta.tagged_sents())
```

```
71 dataset2 = [[w[0] for w in sent] for sent in nltk.corpus.mac_morpho.
       tagged_paras()]
72
73 traindata = [[(w, convert_to_universal_tag(t)) for (w, t) in sent] for sent in
       dataset1]
74 traindata2 = traindata + [[(w, convert_to_universal_tag(t, reverse=True)) for
       (w, t) in sent] for sent in dataset2]
75
76 shuffle(traindata)
77 shuffle(traindata2)
78
79 # 10 por cento para teste
80 test_data1_length = int(len(traindata) * 0.1)
81 test_data2_length = int(len(traindata2) * 0.1)
82 test_data = traindata[:test_data1_length] + traindata2[:test_data2_length]
83 traindata = traindata[test_data1_length:]
84 traindata2 = traindata2[test_data2_length:]
85
86 regex_patterns = [
87     (r"^[nN][ao]s?$", "ADP"),
88     (r"^[dD][ao]s?$", "ADP"),
89     (r"^[pP]el[ao]s?$", "ADP"),
90     (r"^[nN]est[ae]s?$", "ADP"),
91     (r"^[nN]um$", "ADP"),
92     (r"^[nN]ess[ae]s?$", "ADP"),
93     (r"^[nN]aquel[ae]s?$", "ADP"),
94     (r"^\xe0$", "ADP"),
95     (r"^[cC]elular(es)?$", "NOUN"),
96 ]
97
98 tagger = nltk.BigramTagger(
99     traindata, backoff=nltk.RegexpTagger(
100         regex_patterns, backoff=nltk.UnigramTagger(
101             traindata2, backoff=nltk.AffixTagger(
102                 traindata2, backoff=nltk.DefaultTagger('NOUN')
103             )
104         )
105     )
106 )
107 templates = nltk.brill.fntbl37()
108 tagger = nltk.BrillTaggerTrainer(tagger, templates)
109 tagger = tagger.train(traindata2, max_rules=100)
110
111 print('Acuracia do pos tagger: {0}'.format(tagger.evaluate(test_data)))
112
113 with open("tagger.pkl", "wb") as f:
114     pickle.dump(tagger, f)
```

APÊNDICE C - Artigo desenvolvido
Solução computacional para classificação e sumarização de polaridade de comentários em português

Irenio Lima Jesus de Aragão[1] **, Nildo Wilpert Júnior**[1]

[1]Departamento de Informática e Estatística – Universidade Federal de Santa Catarina (UFSC)
Florianópolis – SC – Brasil

{irenio.limaj,nildowjunior}@gmail.com

Resumo. *Analisar a qualidade de um produto ou serviço levando em consideração os comentário de outros consumidores se tornou uma prática comum nos dias de hoje. Realizar esta análise de forma consolidada, eficiente e confiável através de um conjunto de comentários é, muitas vezes, impraticável. Programas capazes de auxiliar os usuários a realizar tal tarefa se fazem necessários. Tanto para auxiliar consumidores, quanto organizações, que veem no feedback de seus clientes um meio para melhorar seus produtos e serviços. O objetivo deste trabalho é propor uma solução computacional capaz de realizar a análise de um conjunto de comentários, em português do Brasil, com o intuito de levantar os principais pontos positivos e negativos sobre um determinado produto, apresentando-os em forma de sumário estatístico. Para tal propósito, realizou-se um estudo sobre as técnicas de inteligência artificial e processamento de linguagem natural empregadas na solução deste tipo de problema, e implementou-se um protótipo para a solução proposta utilizando tais técnicas.*

Abstract. *Analyze the quality of a product or service taking into consideration the comments of others customers has become a common practice nowadays. Doing analysis on a consolidated, efficient and trustworthy way using a large number of comments is much times impracticable. Computer programs capable of helping users to do this task became necessary, both for customers, as well as organizations, who see feedback of their customers as a way to improve their products and services. The goal of this work is to propose a software capable of analyzing a collection of Brazilian Portuguese comments about one product to show the main positive and negative points about it, presenting them as a statistical summary. For this purpose, a study was carried out on the techniques of artificial intelligence and natural language processing used in the solution of this type of problem, and a prototype for the proposed solution using such techniques was implemented.*

1. Introdução

A opinião de outros consumidores é utilizada como fonte de informação no momento da escolha de um produto há muito tempo. Mas com a popularização da internet, o acesso à essa informação tornou-se cada vez mais facilitado. São poucas as pessoas que adquirem um item sem antes ler comentários sobre o mesmo, com o intuito de saber se estão escolhendo o melhor produto dentro das suas expectativas.

Contudo, o grande número de informação disponível, cujo objetivo inicial é o de compartilhar experiências e auxiliar outros compradores a tomarem a melhor decisão, pode se tornar um problema. Pois analisar um grande conjunto de comentários textuais é demorado e sujeito a subjetividade.

De um lado temos a vantagem da facilidade de acesso a um grande número de dados. Do outro temos a dificuldade para, utilizando estes dados, extrair informações quantitativas que auxiliem a tomada de decisão. Neste cenário percebeu-se a necessidade do desenvolvimento de uma solução computacional capaz de realizar a análise de um conjunto de comentários, escritos em português do Brasil, com o objetivo de determinar as caracteristicas mencionadas pelos usuários, avaliar a polaridade do sentimento em relação as mesmas e gerar um sumário contabilizando estas informações. Para isso utilizaram-se técnicas de PLN bem como de aprendizado de máquina.

As seções a seguir apresentam um breve referencial teórico, seguido de uma análise de trabalhos relacionados, a descrição do modelo proposto e um estudo de caso apresentando o desenvolvimento de um protótipo para tal ferramenta.

2. Processamento de linguagem natural - PLN

O processamento de Linguagem Natural (PLN) pode ser definido como a área da Ciência da Computação responsável por estudar e propor técnicas para o desenvolvimento de programas de computador que sejam capazes de analisar, reconhecer e/ou gerar conteúdo oral ou escrito em linguagens humanas ou linguagens naturais, como o português e o inglês (VIEIRA; LOPES, 2010). Apesar de não ser um ramo de estudo novo, tornou-se ainda mais importante devido ao grande volume de dados disponíveis com o advento da internet. Tendo em vista as diferentes características das linguagens existentes ao redor do mundo, diferentes métodos podem atender às diferentes necessidades de análise e interpretação.

Segundo Jurafsky e Martin (2009) PLN pode ser dividido em seis categorias distintas:

- Análise fonética e fonológica, que trata da relação entre as palavras e seus sons;
- Análise morfológica, responsável pelo estudo da construção das palavras a partir de componentes dotados de significado e da classificação das mesmas em categorias morfológicas;
- Análise sintática, que considera o relacionamento entre as palavras, analisando a estrutura para formação das sentenças;
- Análise semântica, focada no significado das palavras e das sentenças;
- Análise pragmática, que estuda como as sentenças são estruturadas a fim de transmitir uma mensagem entre um locutor e um receptor em um determinado contexto;
- Análise de discurso, cujo objetivo é entender como as sentenças se relacionam para formar estruturas mais complexas.

3. Aprendizado de máquina

Mitchell (1997, p.2, tradução nossa) usa a seguinte definição para aprendizado de máquina: “Diz-se que um programa de computador aprende de uma experiência E, com respeito a uma classe de tarefas T e medida de performance P, se sua performance nas

tarefas de T, mensuradas por P, aumenta com a experiência E"[1]. Ou seja, a capacidade de aprendizado de um sistema ao passar por determinado processo de aprendizagem, é evidenciada mensurando o desempenho do sistema ao executar uma tarefa de seu domínio e comparando a performance anterior e posterior ao processo. Usualmente, aprendizado de máquina é separado em três categorias primárias:

- aprendizado supervisionado, que é caracterizado pela necessidade de um conjunto de dados de treino onde cada entrada é rotulada com uma classe correspondente;
- aprendizado não supervisionado, onde o conjunto de treino não é rotulado, pois o objetivo é a criação de grupos de dados com caracteristicas semelhantes;
- aprendizado semi supervisionado, que aceita conjuntos de treinos parcialmente rotulados. Servindo como opção para cenários onde a obtenção de dados de treino é simples mas o custo para rotulá-los é alto.

Cada uma destas categorias apresenta um conjunto de possíveis algoritmos à serem utilizados e vantagens e desvantages em cenários específicos. A utilização de uma ou outra opção deve ser analisada caso a caso.

4. Mineração de opinião

Análise de sentimento ou mineração de opiniões, do inglês sentiment analysis e opinion mining, respectivamente, "[...] é o campo de estudo que analisa as opiniões, avaliações, atitudes e emoções das pessoas sobre entidades como produtos, serviços, organizações indivíduos, problemas, eventos, tópicos e seus atributos"[2] (LIU, 2012, p.7, tradução nossa).

Técnicas de mineração de opinião podem ser aplicada em textos de diversos tamanhos em diferentes níveis e com diferentes objetivos. Quanto aos níveis a análise pode ser realizada a nível de documento, de sentença e de entidade/aspecto. Algumas das aplicações da mineração de opinião são a extração e sumarização automática de opiniões, integração automática de opiniões de várias fontes, detecção automática de revisões falsas, correção de revisões mal classificada e monitoramento de entidades específicas (BECKER, 2014).

Tsytsarau e Palpanas (2012) diz que diversas técnicas de análise de sentimentos podem ser divididas em três macro-tarefas. São elas: Identificação das features; classificação da polaridade da opinião expressa; e sumarização. Sendo está última a menos complexa, podendo, inclusive, não estar presente em alguns casos ou ser tratada apenas como uma tarefa de pós processamento.

5. Classificação e sumarização de comentários

Para alcançar os objetivos deste trabalho, foi proposto um modelo de ferramenta de classificação e sumarização de comentários textuais sobre produtos e suas características, aqui denominadas *domain features*.

[1] A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E. (**?**, p. 2, tradução nossa)

[2] [...] is the field of study that analyzes people's opinions, sentiments, evaluations, appraisals, attitudes, and emotions towards entities such as products, services, organizations, individuals, issues, events, topics, and their attributes (LIU, 2012, p.7)

Como estrutura geral da ferramenta desenvolvida, foram definidos um fluxo para o processo de treinamento da mesma, representado pela Figura 1, e outro para o seu uso no processo de classificação e sumarização, exemplificado na Figura 2.

O processo de treinamento inicia na etapa 1, onde um *web crawler*(sistema de navegação automática e metódica pela internet, buscando informações pré-definidas em sua configuração) é utilizado para a obtenção de um conjunto de comentários e notas. Estes dados são persistidos em um banco de dados para utilização futura. Na etapa 2 é realizado o pré-processamento dos comentários. Primeiramente é realizada a normalização dos textos (etapa 2.1) incluindo a remoção de links, letras e pontuações repetidas, etc. Os comentários normalizados são separados em sentenças e então são filtrados (etapa 2.2) a fim de evitar o processamento de textos irrelevantes. O conjunto de sentenças que não foram removidas na etapa 2 são enviados para a etapa 3, processamento. Nesta etapa são gerados os *feature sets* (etapa 3.1) contendo as características que serão utilizadas para o treinamento do modelo (etapa 3.2). Na etapa 4 o processo de treinamento gera como saída um modelo de classificação. Este modelo será utilizado posteriormente na etapa 7.3 do processo de classificação e sumarização.

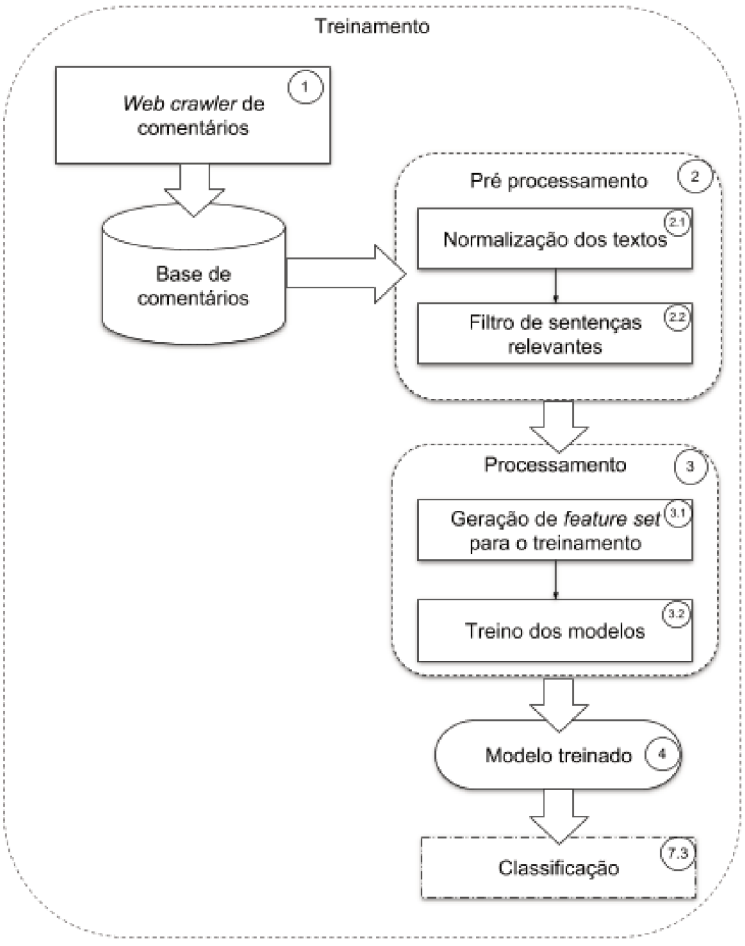

Figura 1. Fluxo básico de treinamento do classificador

O processo de classificação e sumarização inicia na etapa 5, com o conjunto de comentários sobre o qual desejamos realizar a classificação e geração de um sumário. Estes

comentários são persistidos em uma base de dados para possibilitar possíveis análises futuras. Na etapa de pré-processamento (etapa 6) são realizados os mesmos tratamentos da etapa 2 do modelo de treinamento. O conjunto de sentenças que não foi removido na etapa de filtro (etapa 6.2) é processado na etapa 7. Primeiramente é realizada a identificação das domain features presentes na sentença (etapa 7.1). Em seguida são gerados os *feature sets* (etapa 7.2) contendo as características relevantes para a classificação, que é realizada na etapa 7.3 utilizando o modelo resultante da etapa 4 do processo de treinamento. Na etapa 7.4 é realizada a sumarização do resultado da classificação associada a cada domain feature. Na etapa 8 é realizada a geração do sumário de domain features. Esta é a saída final do processo.

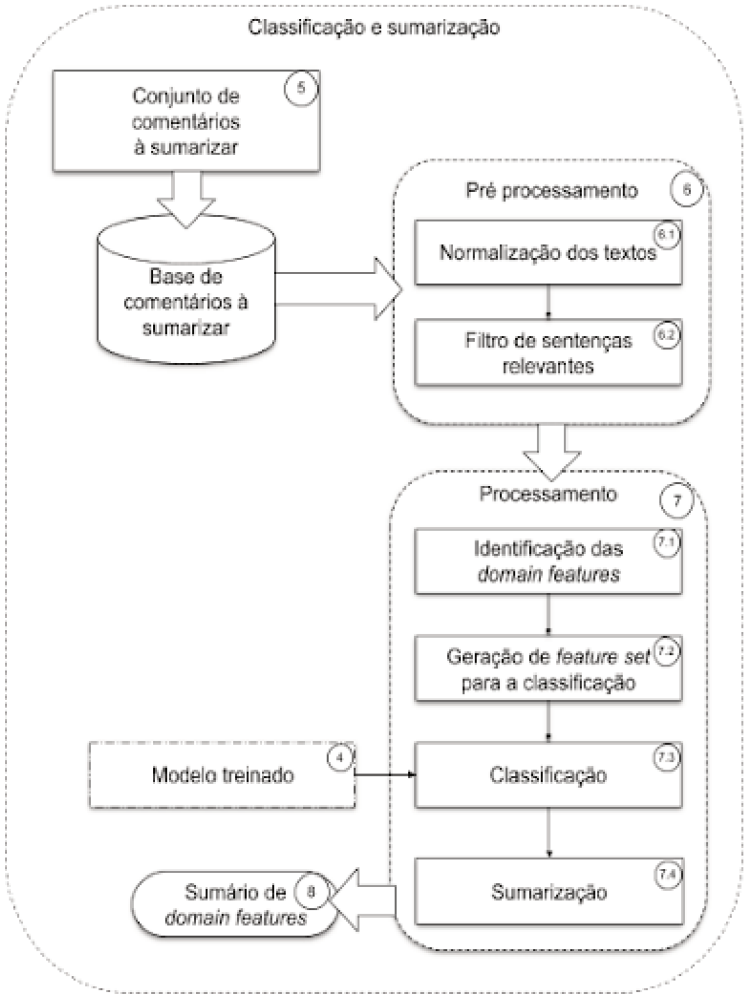

Figura 2. Fluxo básico da classificação e geração do sumário

6. Estudo de caso

Nesta seção serão apresentadas as decisões de projeto bem como o escopo no qual o desenvolvimento da solução proposta está inserido.

Para melhor análise dos resultados, durante o desenvolvimento deste trabalho, considerou-se comentários textuais sobre produtos da categoria de celulares, assumindo que as características pertinentes ao processamento e resultado da ferramenta são apenas aquelas que descrevem um aparelho de telefone celular e sua qualidade.

6.1. Entradas

Para a obtenção dos dados de entrada do sistema desenvolvido, composto por um conjunto de comentários com um nota associada no fluxo de treinamento e um conjunto de comentários sobre um produto especifico no fluxo de classificação e sumarização, fez-se o uso de *web crawlers* preparados para extrair dados do site Buscapé. Deste foram extraídos cerca de 12 mil comentários, que após passarem por um processo de verificação resultaram em 10272 comentários com o conjunto formado por título, conteúdo, data e pontuação únicos.

Antes do uso do Buscapé, o Twitter foi analisado como fonte para obtenção dos comentários, porém notou-se que haviam poucos dados com as características necessárias para o contexto sendo considerado. Foram aplicados filtros na busca dos tweets para obter apenas os que fizessem referência a alguma característica de celular e que contivessem ao menos um emoticon (sequência de caracteres que traduz um estado emotivo), o qual seria utilizado como rótulo para o treinamento do classificador. Com a aplicação dos filtros, pouco mais de 150 tweets foram obtidos, não sendo suficientes para o treinamento e teste de um modelo de classificação.

6.2. Domain Features

Uma lista contendo as *domain features* para a categoria de celulares foi criada de forma manual, analisando características comuns nas descrições de aparelhos celulares. Percebu-se então que algumas das *domain features* selecionadas representavam a mesma característica do aparelho celular. Desta forma, optou-se por agrupar os dados, resultando nas informações contidas na Tabela 1. A utilização de técnicas mais robustas para a identificação dos aspectos presentes nos comentários está prevista para trabalhos futuros.

Tabela 1. Domain features agrupadas por feature principal

***Feature* principal**	***Domain feature***
armazenamento	cartão de memória
	expansível
	memória interna
bateria	autonomia
	tipo de bateria
câmera	câmera frontal
	câmera traseira
dispositivo	aparelho
	celular
	produto
	smartphone
	telefone
multichip	dois chips
	dual chip
sistema operacional	android
	ios
tamanho	altura
	comprimento
	largura
tamanho da tela	tamanho do display
tela	display
	visor
tipo de tela	amoled
	ips
	lcd
	oled
áudio	som
processador	processamento

6.3. Palavras polarizadas

Para obtenção da lista de palavras polarizadas, que é uma das entradas previstas no modelo de classificação deste trabalho, optou-se pela definição manual da mesma. A escolha

das palavras foi baseada em conhecimento de senso comum, selecionando palavras comumente utilizadas para descrever sentimento sobre um objeto sendo avaliado, e seus sinônimos, diretos e indiretos, mais relevantes para o contexto de celulares. A Tabela 2 apresenta as palavras selecionadas.

Tabela 2. Palavras polarizadas selecionadas

Palavras positivas		
acessível	extraordinário	positivo
adorei	feliz	prático
amei	forte	rápida
boa	fácil	rápido
boas	gostei	satisfeito
bom	legal	superior
bonito	magnífico	surpreendente
contente	maravilhosa	ótima
durável	maravilhoso	ótimo
eficiente	melhor	útil
excelente	perfeito	

Palavras negativas		
desagradável	incorreto	miserável
desprezível	inferior	má
detestável	insatisfatório	negativo
errado	insatisfeito	ordinário
fajuto	insuportável	pior
feio	inúteis	podre
fraca	inútil	péssima
fraco	lamentável	péssimo
horrível	lento	ruim
imprestável	maldito	terrível

6.4. Classificação

O modelo proposto considera o uso de técnicas de classificação por aprendizado supervisionado. Essas técnicas são implementados por diversas ferramentas de PLN, porém, no escopo deste trabalho decidiu-se por utilizar a toolkit NLTK. Dentro os diversos algoritmos existentes para o treinamento do classificador optou-se pela utilização dos seguintes: MaxEnt GIS (Generalized Iterative Scaling), MaxEnt IIS (Improved Iterative Scaling), Naive Bayes e Decision Tree. Isto foi feito com o objetivo de possibilitar a comparação dos resultados.

Os mesmos subconjuntos de treino e teste foram utilizados para cada classificador. Na definição de tamanho desses subconjuntos, priorizou-se o treino, buscando submeter um maior número de exemplos na etapa de aprendizagem, para melhorar a acurácia do classificador. Além disso, o conjunto de dados obtido possui uma grande quantidade de exemplos (10272 comentários). Portanto, o conjunto de comentários foi dividido em 90% (9245) para treino e 10% (1027) para teste.

7. Protótipo

Para exemplificar e avaliar a proposta, implementou-se um software como protótipo da ferramenta, cuja entrada é um conjunto de comentários textuais, escritos em português do Brasil, sobre aparelhos celulares e a saída é um documento, em formato de sumário, listando as *domain feature* com o total de comentários que classificaram tal característica de forma positiva e negativa.

7.1. Entradas

O protótipo desenvolvido utiliza como entrada principal dois formatos de dados distintos, um para o processo de treinamento: um conjunto de tuplas compostas de comentários textuais, escritos em português do Brasil, e uma nota numérica, de 1 a 5; e outro para o

processo de classificação e sumarização: apenas um conjunto de comentários escritos em português do Brasil.

Comentários: para a obtenção dos comentários foi utilizado um *web crawler* para realizar o *scraping* (extração de dados) de comentários de usuários sobre aparelhos celular no site Buscapé, escolhido entre as diversas opções disponíveis por agrupar informações provenientes de um grande conjunto de outros sites. Foram obtidos 10272 comentários, os quais foram armazenados em um banco de dados e utilizados nas etapas seguintes.

Domain features: após avaliação da ferramenta DBPedia Spotlight (uma base de dados estruturados), onde ela não se mostrou preparada para identificar as características do contexto de celulares, optou-se pela captura manual das características mais relevantes e suas variações.

Palavras polarizadas: as palavras que denotam sentimento positivo ou negativo foram obtidas através de um conjunto inicial baseado em conhecimento de senso comum (palavras como bom, ótimo, ruim, péssimo) e de busca por sinônimos de forma manual e recursiva em dicionários online a partir deste conjunto inicial.

7.2. Pré-processamento

A etapa de pré-processamento engloba as etapas de normalização do texto para o processamento, divisão dos comentários em sentenças e uma pré filtragem dos comentários. Estas sub etapas são comuns aos processos de treinamento e de classificação e sumarização.

Normalização dos dados: nesta etapa são realizadas sobre os comentários as tarefas responsáveis por transformar o texto para um formato padrão, sendo elas (1) a conversão para letras minúsculas; (2) a remoção de URLs; (3) a conversão de quebras de linha em pontuação; (4) a correção de erros gramaticais comuns; e (5) a remoção de espaços e pontuações duplicadas.

Divisão em sentenças: considerando que o modelo proposto prevê a análise dos comentários a nível de sentença, utilizou-se um modelo desenvolvido em python e disponibilizado pela biblioteca NLTK, treinado para divisão de sentenças do português do Brasil.

Filtro de sentenças inicial: esta etapa de filtragem leva em consideração o tamanho da sentença, descartando as que possuem menos de 3 caracteres, e a presença de ao menos uma *domain feature*, descartando sentenças que não contenham ao menos um dos valores presentes na lista de *domain features* obtida anteriormente.

7.3. Processamento

A etapa de processamento inclui as tarefas de manipulação das sentenças, utilizando técnicas de PLN, a fim de permitir o treinamento e a posterior classificação de novas entradas. Assim como a etapa de pré-processamento, esta etapa possui sub etapas comuns entre o fluxo de treinamento e o fluxo de classificação e sumarização. Estas sub etapas são:

Tokenização: a tokenização consiste da divisão da sentença em tokens, ou seja, o conjunto de elementos que compõem a mesma. No protótipo desenvolvido, esta tarefa foi realizada utilizando o tokenizador disponibilizado pela biblioteca NLTK, configurado para o idioma português do Brasil.

POS tagging: esta tarefa atribui a cada token um rótulo que representa a sua classe gramatical dentro da sentença. Para tal propósito, utilizou-se um *tagger open source* criado pela comunidade, chamado Nltk-Tagger-Portuguese.

Filtro de sentenças: nesta etapa é verificado se no conjunto de tokens que compõem a sentença há a presença de ao menos um dos itens provenientes das listas de palavras de sentimentos (palavras polarizadas). Além disso, o sistema verifica se sentenças consideradas positivas (nota maior ou igual a 3) possuem alguma palavra de sentimento negativa e se sentenças consideradas negativas (nota menor que 3) possuem alguma palavra de sentimento positiva, o que indica uma possível contradição.

Geração de *feature set*: *feature sets* são conjuntos de dados derivados das sentenças, que serão utilizados para o treinamento do classificador e posterior classificação de novas sentenças. No protótipo desenvolvido, os *feature sets* são compostos por três fatores, sendo eles (1) a presença de palavras com polaridade positiva; (2) presença de palavras com polaridade negativa; e (3) presença de palavras de negação próximas as palavras de sentimentos.

Treinamento do classificador: essa etapa é realizada utilizando o conjunto de feature sets associados às suas classes, e os algoritmos de aprendizagem MaxEnt GIS, MaxEnt IIS, Naive Bayes e Decision Tree. O conjunto de comentários, antes obtido e armazenado localmente, foram submetidos ao pré-processamento e divididos em 2 subconjuntos, um para treino (90% dos comentários) e outro para teste (10% dos comentários) dos modelos treinados. Os modelos gerados são então utilizados na classificação das sentenças.

Classificação: nesta etapa cada um dos modelos treinados é utilizado, em conjunto dos *feature sets* derivados das sentenças, para classificar essas sentenças em uma das classes propostas (positivo ou negativo).

Sumarização: neste passo são identificadas as *domain features* presentes nas sentenças classificadas. É realizada a contagem dos comentários de forma agrupada por *domain feature* e polaridade. O sumário final é gerado contendo cada *domain feature* mencionada, seguida do total de avaliações positivas e negativas sobre ela. Estes dados são ordenados de forma decrescente pelo somatório das avaliações.

7.4. Avaliação

A avaliação do protótipo implementado direcionou-se para a análise da veracidade do sumário gerado pela ferramenta, comparando-o à lista de comentários com as características identificadas e classificadas por um humano, aqui denominado sumário ideal. Foram selecionados aleatoriamente 100 comentários, os quais foram analisados pelos autores para gerar o sumário ideal, e submetidos como entrada para o protótipo desenvolvido, gerando os sumários a serem avaliados (um para cada modelo de classificação utilizado).

A avaliação dos sumários gerados pelo protótipo foi realizada utilizando as métricas utilizadas em Salas-Zárate (2017), sendo elas:

$$Precision = \frac{VerdadeirosPositivos}{VerdadeirosPositivos + FalsosPositivos}$$

$$Recall = \frac{VerdadeirosPositivos}{VerdadeirosPositivos + FalsosNegativos}$$

$F - score = 2 * \frac{Precision * Recall}{Precision + Recall}$

Assim como em Salas-Zárate (2017), as métricas foram utilizadas sobre dois aspectos do protótipo: (1) a capacidade de identificar características no texto e (2) a capacidade de identificar a polaridade do sentimento expresso no texto sobre cada característica.

7.5. Resultados

Aplicando-se o método apresentado na seção 7.4 ao protótipo desenvolvido, foram obtidos os resultados das Tabelas 3 e 4, que apresentam os resultados quanto à precisão na identificação de características e à precisão na classificação de polaridade, respectivamente.

Tabela 3. Precision, recall e F-score das características identificadas pelo sistema

Precision	Recall	F-score
0,737	0,438	0,549

A Tabela 3 indica que, de cada 100 características identificadas pelo sistema, 74 realmente estão presentes e são alvos de sentimento no texto (precision). Por outro lado, indica que o sistema identificou menos da metade das características alvos de sentimento presentes no texto (recall).

Tabela 4. Precision (P), recall (R) e F-score (F) das polaridades indicadas pelo sistema

Naive Bayes			Decision Tree			MaxEnt GIS			MaxEnt IIS		
P	R	F	P	R	F	P	R	F	P	R	F
0,41	0,62	0,48	0,41	0,62	0,48	0,41	0,62	0,48	0,41	0,62	0,48

A Tabela 4 mostra que a única diferença significativa entre os algoritmos foi a vantagem de 0,17 na acurácia da classificação com Decision Tree. Além disso, indica que 62% das polaridades verdadeiramente positivas, foram classificadas corretamente pelo sistema (recall); e apenas 41% do total de ocorrências que o sistema classificou como positivas eram realmente positivas (precision).

8. Conclusão

Com os resultados obtidos, nota-se que (1) a efetividade da identificação de características teve grande influência sobre os resultados da classificação de polaridade, pois a classificação é feita apenas sobre as características identificadas, analisando as sentenças em que essas são referenciadas; (2) a acurácia do classificador, que, apesar de não ter sido baixa para nenhum dos algoritmos, não foi suficiente para atingir uma boa precisão no resultado final do sistema; (3) a abordagem de definir uma lista fixa de características a serem reconhecidas pelo sistema (domain features), limita a abrangência da etapa de extração de características do produto a partir do texto.

Por fim, conclui-se que o modelo proposto tem potencial para solução do problema identificado. Apesar de não ter atingido um bom nível de confiabilidade, ele apresenta um resumo qualitativo dos comentários processados, refletindo com certa precisão a

opinião dos consumidores e servindo como auxílio em decisões de compra ou estratégias de mercado. Os resultados obtidos foram, no geral, medianos, mas encorajadores para a evolução do modelo em trabalhos futuros.

8.1. Trabalhos Futuros

Considerando o potencial do modelo proposto para a solução do problema identificado, os resultados deste trabalho e as limitações evidenciadas por esses, foram identificados algumas oportunidades de evolução do modelo. As possíveis melhorias são listadas a seguir, como sugestões para trabalhos futuros.

Alterar a forma como os comentários são divididos em unidades menores, criando unidades de sentimento através da análise morfológica e sintatica das palavras próximas de cada característica do produto identificada no texto. Assim, seriam descartadas as palavras que não expressam sentimentos e nem modificam as que expressam.

Utilizar técnicas mais abrangentes para a correção automática das palavras contidas nos comentários, como técnicas baseadas em dicionário, que tratam um maior número de palavras e possíveis erros sobre elas.

Aumentar a cobertura da identificação de características do produto, gerando de forma automática a lista de características pré-definidas, analisando estatística e semânticamente o conjunto de dados para identificar as características citadas ou consultando bases de informação estruturada (alternativas ao DBPedia Spotlight).

Utilizar uma fonte de dados com maior confiabilidade da relação entre o texto e a classificação do mesmo, para evitar a incoerência apresentada em alguns comentários observados no estudo, onde o conteúdo se mostrava positivo, mas a nota aplicada pelo avaliador era negativa e vice versa. Para tal propósito, pode-se, por exemplo, realizar a classificação manual de um conjunto expressivo de comentários, criando uma base de dados para treinamento.

Expandir o conjunto de palavras que o modelo considera como expressões de sentimentos, utilizando léxicos de sentimentos ou identificando expressões de sentimento através de características textuais da sentença (morfologia e a semântica).

Aumentar o ganho de informação com o feature set da classificação de polaridade, considerando também características estruturais da sentença.

Utilizar outras técnicas de mineração de opiniões e PLN nas etapas do modelo, realizando novos testes e avaliando a evolução dos resultados. Exemplos de outras técnicas são: Pointwise Mutual Information (PMI), Semantic-orientation Latent Semantic Analysis (SO-LSA), Latent Dirichlet Allocation (LDA) e Lematização.

Referências

BECKER, Karin. **Introdução à Mineração de Opiniões**. In: Salgado, A.C.; Lóscio, B.F.. (Org.). Atualizações em Informática / XXXIV Congresso da Sociedade Brasileira de Computação. 1ed.Porto Alegre: SBC, 2014, v. 1, p. 125-175.

JURAFSKY, Daniel; MARTIN, James H. **Speech and Language Processing**: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. v. 21, p. 0–934, 2009.

LIU, Bing; ZHANG, Lei. A survey of opinion mining and sentiment analysis. In: AGGARWAL, Charu C.; ZHAI, Chengxiang. **Mining text data**. New York: Springer US, 2012. Cap. 13. p. 415-463.

MITCHELL, T. **Machine Learning**. New York, NY: McGraw-Hill, 1997. ISBN 0-07-042807-7.

SALAS-ZÁRATE, María Del Pilar et al. Sentiment Analysis on Tweets about Diabetes: An Aspect-Level Approach. **Computational and Mathematical Methods in Medicine**, [s. l.], v. 2017, 2017.

TSYTSARAU, M.; PALPANAS, T. **Survey on mining subjective data on the web**. Data Mining and Knowledge Discovery, v. 24, n. 3, p. 478–514, May 2012. ISSN 1573-756X.Disponível em:<http://dx.doi.org/10.1007/s10618-011-0238-6>.

VIEIRA, Renata; LOPES, Lucelene. Processamento de linguagem natural e o tratamento computacional de linguagens científicas. In: PERNA, Cristina Lopes; DELGADO, Heloísa Koch; FINATTO, Maria José (Org.). **Linguagens especializadas em corpora:** Modos de dizer e interfaces de pesquisa. Porto Alegre: Edipucrs, 2010. p. 183-201. (ISBN 978-85-397-0024-0). Disponível em: <http://www.pucrs.br/edipucrs/linguagensespecializadasemcorpora.pdf>. Acesso em: 19 ago. 2016.